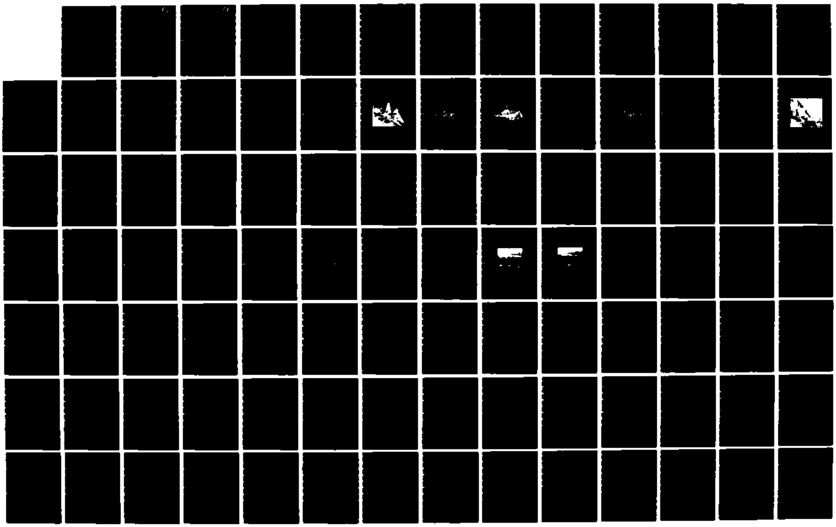
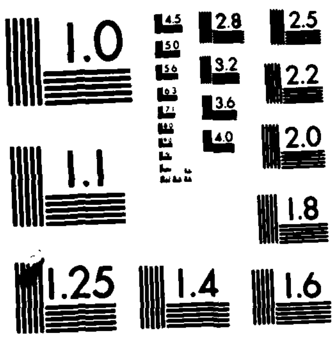
AD-A138 421 IMAGEFFILTERING WITH BOOLEAN AND STATISTICAL OPERATORS 1/3
(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH
SCHOOL OF ENGINEERING R D WELLS DEC 83
UNCLASSIFIED AFIT/GE/EE/83D-72 F/G 20/6 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ADA138421

IMAGE FILTERING WITH
BOOLEAN AND STATISTICAL OPERATORS

THESIS

AFIT/GE/EE/83D-72

ROBERT D. WELLS
Capt USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

84 02 27 059

AFIT/GE/EE/83D-72

Accession For	
NTIS GRA&I	X
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

IMAGE FILTERING WITH BOOLEAN AND STATISTICAL OPERATORS

THESIS

AFIT/GE/EE/83D-72 ROBERT D. WELLS
Capt USAF

Approved for public release; distribution unlimited

AFIT/GE/EE/83D-72

IMAGE FILTERING WITH
BOOLEAN AND STATISTICAL OPERATORS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

Robert D. Wells, B.S.
Capt USAF

Graduate Electrical Engineering

December 1983

Approved for public release; distribution unlimited

ACKNOWLEDGEMENTS

I deeply appreciate the excellent guidance from my thesis advisor, Dr Matthew Kabrisky. My thesis committee members, Maj Kenneth Castor and Capt David King, aided my research with their expertise. I also thank the System Manager and Engineer of the Signal Processing Laboratory, Maj Larry Kizer and Dan Zambon, for their support. Finally, I acknowledge Karen Moore for typing this.

CONTENTS

Page

LIST OF FIGURES

ABSTRACT

Edge extraction is an image processing technique for defining the edge information in an image. This effort researches different edging processes as applied to preprocessing for two pattern recognition processes. The first one is a cross-correlation method to find a target given that the target has a known size, orientation, and aspect. Correlation is performed in the spatial frequency domain with two-dimensional fast Fourier transforms of the searched edge image and a hand drawn edge template to correct for translation only.

The second pattern recognition process researched also uses edging as one step of a purely spatial domain algorithm. The approach locates targets in infrared images that can be described a "hot" clusters. A cluster recognition algorithm by Hamadani is implemented and altered for testing of local thresholding and thresholding rules. The algorithm is shown to be effective on real infrared images, provided by the thesis sponsor, the Air Force Armament Laboratory at Eglin AFB, Florida.

IMAGE FILTERING WITH

BOOLEAN AND STATISTICAL OPERATORS

I. INTRODUCTION

BACKGROUND

An image is the representation of the electro-magnetic energy at a certain time in a plane in space. Typically the energy is in the spectrum of visible light or infrared light. Image pattern recognition is the process of analyzing an image to find known patterns that aid in the classification of the image information content. Image pattern recognition has many applications such as: (Ref 1)

1) document processing
2) industrial automation
3) medicine and biology
4) military target finding, guidance.

There is no pattern recognition machine to date that approaches human capabilities. Pattern recognition problems are difficult because of the natural complexity of the world. Humans use form (gestalt) matching whereas all machines perform template matching when the form has a specific rendition. Machines can work practically under controlled conditions. For example, retail stores often use bar code labels on their products for inventory control. Lasers can read these carefully printed labels if they are held at the proper distance.

The general approach for recognition of objects is to extract a set of features that describe the object and search the scene for a match. The set of features is the template and the searching

process is called template matching. The selection of features will determine the success of the match. (Ref 2)

The mechanics of the matching process must employ some decision theory that will allow for the unavoidable errors. The acceptance region for a particular feature class should be as large as possible without overlapping the region for a different class. Often a confidence rating will be assigned to each match.

Segmentation is the process of dividing the image into meaningful parts during the analysis. Meaningful parts might be sections of the scene where there is a likelihood of finding the sought objects; or they might be regions that correspond to parts of the sought object.

When a process has the ability to determine how to beneficially change its version of the template, then we say that it has learning ability. Learning should improve the feature set or it will fail. Artificial intelligence, a field similar to pattern recognition, lends much to learning theory.

SCOPE

This thesis works with digitized images of visible light and infrared light quantized into 16 grey levels. The video, or visible light, images are represented in 256 X 256 matrices, while the infrared images are represented in 96 X 100 matrices. The video images were digitized in-house from photographs. The Air Force Armament Laboratory, Eglin AFB, Florida, provided the infrared images. These images portray real scenes with military application.

Images may be described by many features, but common ones are edges, texture, regions, shapes, and spatial frequency content. The first four listed are defined in the spatial domain, while the last is defined by a transformation such as the Fourier transform.

Preprocessing is the filtering or reduction of an image before the template matching process. This thesis is concerned mainly with the use of linear and nonlinear spatial filters for the preprocessing of an image. Especially, the filters will be designed to extract the edges in the image. This is similar to high frequency enhancement in the frequency domain.

Some template matching processes are considered, and the edge extraction is optimized for each template matching process. While edge extraction is emphasized, no fitting of mathematical models for texture or shapes is used.

The techniques to be used will be chosen to be as immune as possible to clutter. Clutter is all of the background scene that distracts the recognition process from finding the object. Clutter may also be another object that occludes the sought object because of their relative positions.

Present applications of image pattern recognition work under constraints that depend on a prior knowledge of the situation at hand. The general problem does not assume these constraints. This thesis does not attempt to solve the complete general problem, rather it researches techniques that are hoped to contribute to the solution of the general problem.

While new techniques are tried on real scene images, the results are visually inspected for information content. The information content in question is in the edges of the image. Visual inspection and careful observation replace any mathematical development to explain what exactly the process did. Hypotheses of beneficial operations are developed without regard to linearity or causality.

This effort assumes the perspective of the target is fixed and known. Therefore, geometrical transformations or stereomapping are not considered. Segmentation techniques are also neglected. Furthermore, the analysis is static, i.e.: successive images are not compared. Hence, no learning ability, movement detection, or three dimensional analysis is discussed.

APPROACH

Several edge extraction or edging algorithms are evaluated for their representation of important image information content. The definition of important information is subject to the data set, and to the intended processing subsequent to the edging process. In general, edges are extracted from areas with monotonically changing brightness over a small area (typically 3 X 3 mask neighborhood). The edges should also be optimized for the template matching algorithm used in the recognition process. Modifications to the evaluated edging algorithms are tried. The resulting images are examined again for important image information content.

Clutter or noise reduction is in the form of thresholding. The thresholding is typically some boolean function based on statistical operators. The statistics may be global or local. Global thresholds depend on the statistics of the entire image, whereas local thresholds depend on a local neighborhood of the pixel being processed.

Edged image values range from zero (representing no edge) to fifteen (representing the highest contrast edge). When edge image pixels are thresholded, the resulting value is either zero or non-zero. A value of zero precludes further processing for that pixel. A non-zero valued pixel after thresholding can take on its original edge value or some function thereof.

The Hamadani algorithm is implemented and evaluated for techniques that can be applied to more general pattern recognition problems. (Ref 3) The Hamadani algorithm works solely in the spatial domain

for the purpose of detecting "hot" clusters in an infrared image. Modifications to the Hamadani algorithm are tested to see if target detection can be improved. Among the proposed modifications are improved edging algorithms, local thresholding, and faster implementation techniques.

A more complicated template matching process than cluster detection is used to operate on edged images to test the effectiveness of the edging algorithms. (Ref 4) Cross-correlation in the frequency domain with two-dimensional fast Fourier transforms (2DFFT) should correct template translation. Cromer concludes that scene brightness cannot be used successfully in this template matching process, and recommends preprocessing with edging routines.

New template matching processes are contemplated to more effectively use edged image information. Proposed techniques are discussed as possible subjects of more research.

EQUIPMENT

All thesis work is done in the Signal Processing Laboratory at the Air Force Institute of Technology. All software is written in Fortran 5 to be run on the Eclipse S/250 computer system. The system has complete video input and output facilities. (Ref 5) Output can be achieved on a video monitor or a Printronix 300 line-printer. The lineprinter output is used to represent images for inclusion in this thesis. Input from a vidicon camera and output to the video monitor are executed on the Nova/Cromenco system which shares disk memory with the Eclipse S/250 system. Support software routines are included in Appendix A.

II. EDGING ALGORITHMS

BOOLPASS

At the start of this thesis effort, it was believed that there was some merit to a proposed edging routine devised by Blaine Feltmate (Ref 6). The operation, entitled Boolpass, consists of three basic steps:

1) average or low pass filter the original image

2) negate the averaged image

3) threshold using the criteria that the original image and the negated averaged image must be within a small range of each other on the gray scale.

Boolpass is almost equivalent to thresholding the original image around the middle of the gray scale. The operation does a selective thresholding of the middle of the gray scale, though. That is, only areas where the average image is in the middle of the gray scale remain after thresholding.

For certain pictures, the operation tends to reduce the image to important edges. Important edges are those edges that are critical to the definition of the target. The operation left many areas containing no edges unreduced, however. It also eliminated some edges that were present in some areas that had no values in the middle of the gray scale.

Many variations were tried on Boolpass as suggested by Feltmate, and some others, too. Nothing seemed to be able to leave all important edges and reduce unimportant areas. Then, using the criterion that deviation for the neighborhood around the pixel

being operated on must be over a certain value, Boolpass gained the ability to reduce areas with middle scale values and with no edges. However, edges not in the middle of the gray scale still were not present. I then concluded that Boolpass was no longer viable and pursued the idea that deviation was the feature that defined edges.

DEVIATION

Deviation works well at defining the edges (see Figures 1 and 2). Specifically, mean deviation was used instead of standard deviation because of faster computation and noise immunity. Large masks, or neighborhoods, tend to blur the edges (see Figure 3). Therefore, most edge processing uses 3 X 3 masks.

Appendix B contains the listings of the Fortran programs used for edging. The program used to calculate the averages and mean deviation on local neighborhoods is STATISTICS.

While deviation works well at defining the edges, random noise at some parts of the image that contain no edges is also extracted to be part of the resulting image. We do not want noise to be part of an edged image.

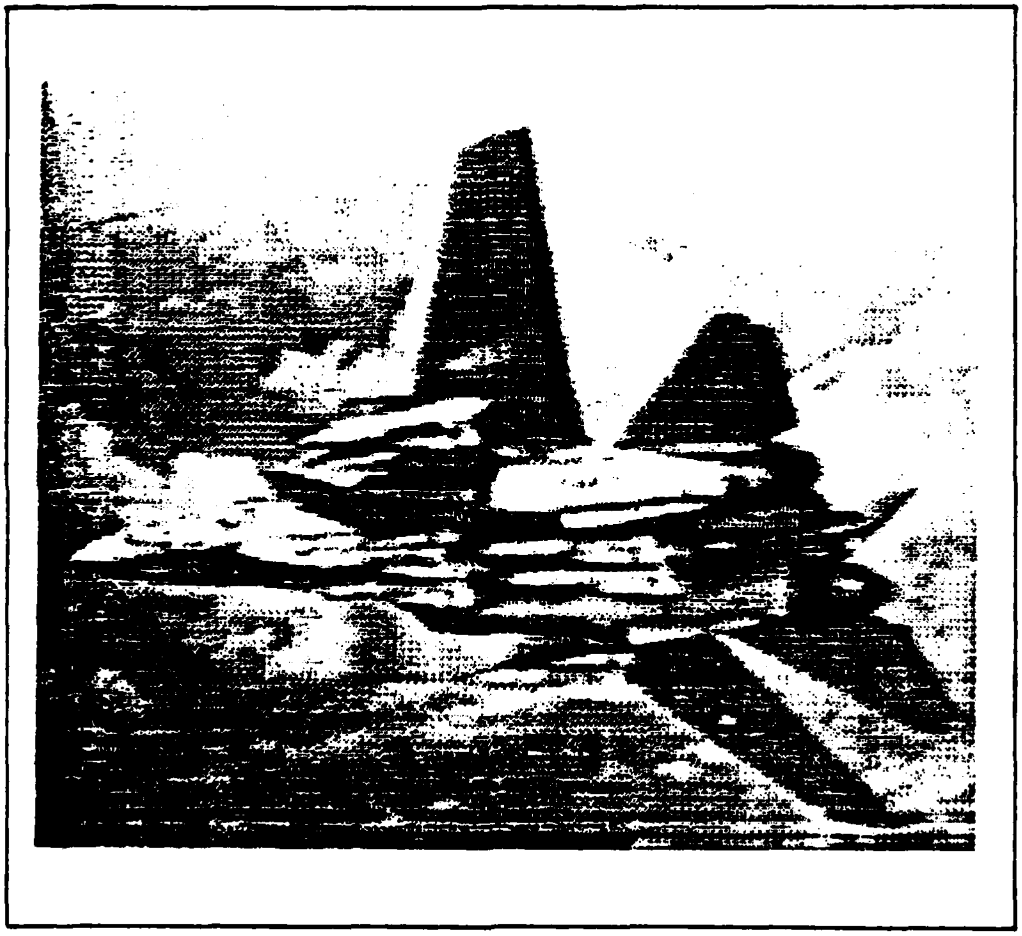


Figure 1 - Original F14 Tomcat

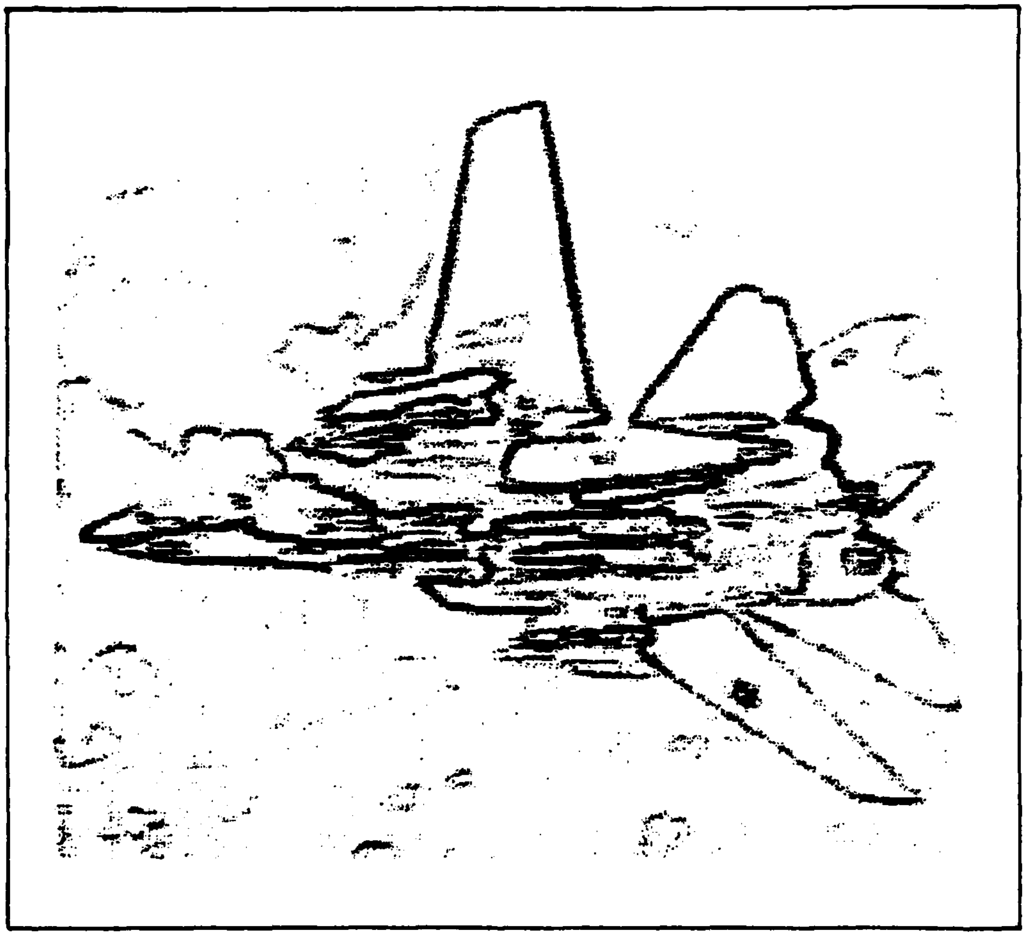


Figure 2 - Deviation Edging of F14 on 3 X 3 Mask

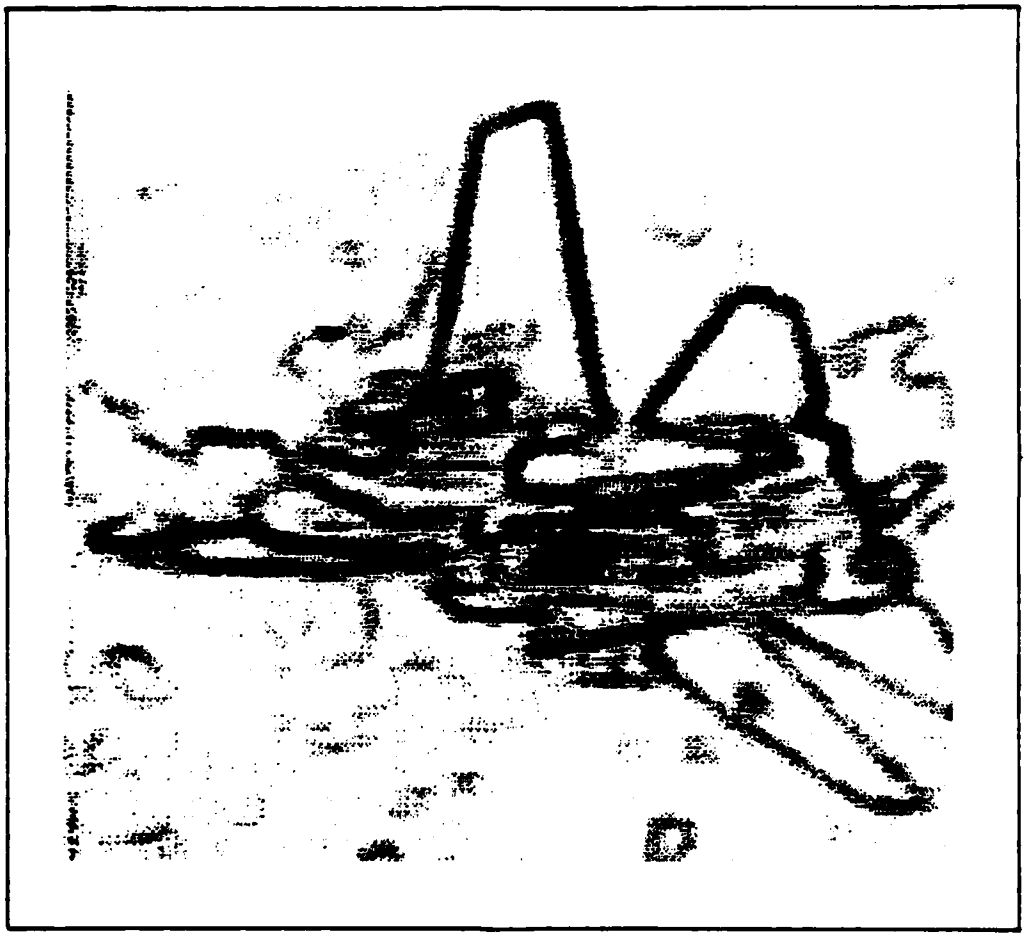


Figure 3 - Deviation Edging of F14 on 7 X 7 Mask

MASK EDGING

Kirsch's operator was tried because of recommendation by Hamadani (Ref 3). Coefficients were changed to effectively use the gray scale range, but relative ratios remain the same (5 to 3). There are eight mask orientations to be convolved with the neighborhood of each pixel. The maximum magnitude of the eight convolutions is assigned to the pixel in the edged image. That maximum occurs for the orientation that most closely aligns with the edge in the original image. The result of performing Kirsch's operator on Figure 1 is shown in Figure 4. The Fortran program for the Kirsch operator is KEDGE, and the listing is in Appendix B.

Mask edging is the process of convolving several masks with the neighborhood of the pixel being processed (Ref 7:97). The masks represent model edge neighborhoods at various orientations. The maximum convolution of the masks becomes the edge value of the output pixel. Notice that the orientation of the edge will be known after the maximum is found.

A mask edging operator, henceforth referred to as the wedge operator, was defined on a 3 X 3 mask with coefficients adding to zero so that the output will be zero for a pixel that is not near an edge. Figure 5 shows the masks for the wedge operator at the four orientations. The actual coefficients are less but the relative ratios remain the same (2 to 1).

Let us look at the masks for orientation #1 of the wedge operator. These masks are models of horizontal edges where the center pixel is:

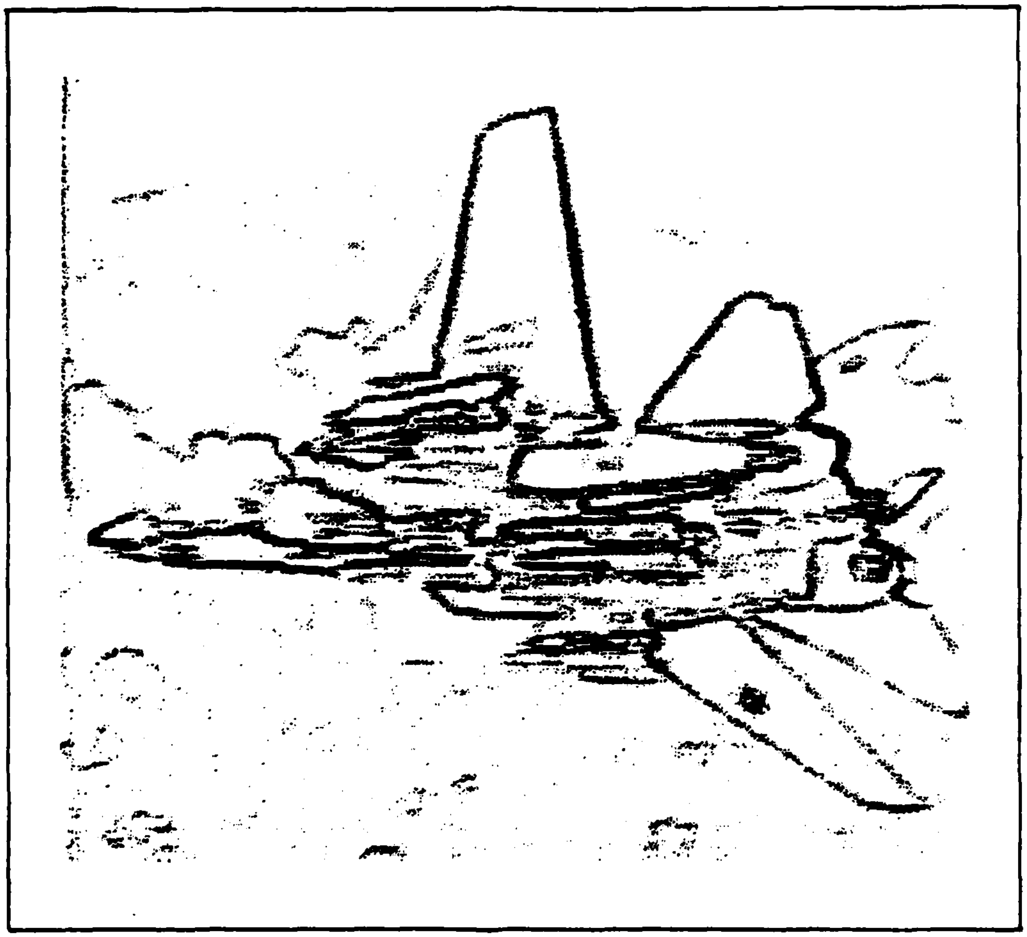


Figure 4 - Kirsch Edging of F14

Orientation #1:

$$\begin{array}{|c|c|c|}\hline 2 & 2 & 2 \\ \hline -1 & -1 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & -1 & -1 \\ \hline 2 & 2 & 2 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline -2 & -2 & -2 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline -2 & -2 & -2 \\ \hline \end{array}$$

Orientation #2:

$$\begin{array}{|c|c|c|}\hline -1 & 2 & 2 \\ \hline -1 & -1 & 2 \\ \hline -1 & -1 & -1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline 2 & -1 & -1 \\ \hline 2 & 2 & -1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline 1 & -2 & -2 \\ \hline 1 & 1 & -2 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline -2 & 1 & 1 \\ \hline -2 & -2 & 1 \\ \hline \end{array}$$

Orientation #3:

$$\begin{array}{|c|c|c|}\hline -1 & -1 & 2 \\ \hline -1 & -1 & 2 \\ \hline -1 & -1 & 2 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline 2 & -1 & -1 \\ \hline 2 & -1 & -1 \\ \hline 2 & -1 & -1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline 1 & 1 & -2 \\ \hline 1 & 1 & -2 \\ \hline 1 & 1 & -2 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline -2 & 1 & 1 \\ \hline -2 & 1 & 1 \\ \hline -2 & 1 & 1 \\ \hline \end{array}$$

Orientation #4:

$$\begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & -1 & 2 \\ \hline -1 & 2 & 2 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline 2 & 2 & -1 \\ \hline 2 & -1 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & -2 \\ \hline 1 & -2 & -2 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline -2 & -2 & 1 \\ \hline -2 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Figure 5 - Convolution Masks for the Wedge Operator

1) negative (relatively) with negative values below

2) negative with negative values above

3) positive with positive values below

4) positive with positive values above

Orientations #2 and #4 are opposing diagonals and #3 is vertical. Notice that half of the masks are negatives of the others.

In order to save the information about which mask orientation produced the maximum convolution value, the wedge operator separates the output into four different images. Each output image is zero filled initially. If the mask with maximum convolution value is at orientation #1, the edge pixel value will be assigned to edge image #1. Edge pixels at orientation #2 are assigned to edge image #2. Similarly, edged images #3 and #4 are formed. If two or more masks at different orientations produce the same maximum convolution value, the edge output goes to each of their corresponding edge images.

Other methods to save the orientation information can be devised to be more efficient. The wedge operator separates the edge images for use in a scheme for correlation described later. The program for the wedge operator, WEDGE, is listed in Appendix B. A program used to recombine the edge images is NCOMB in Appendix A. Figures 6-11 show results of the wedge operator.

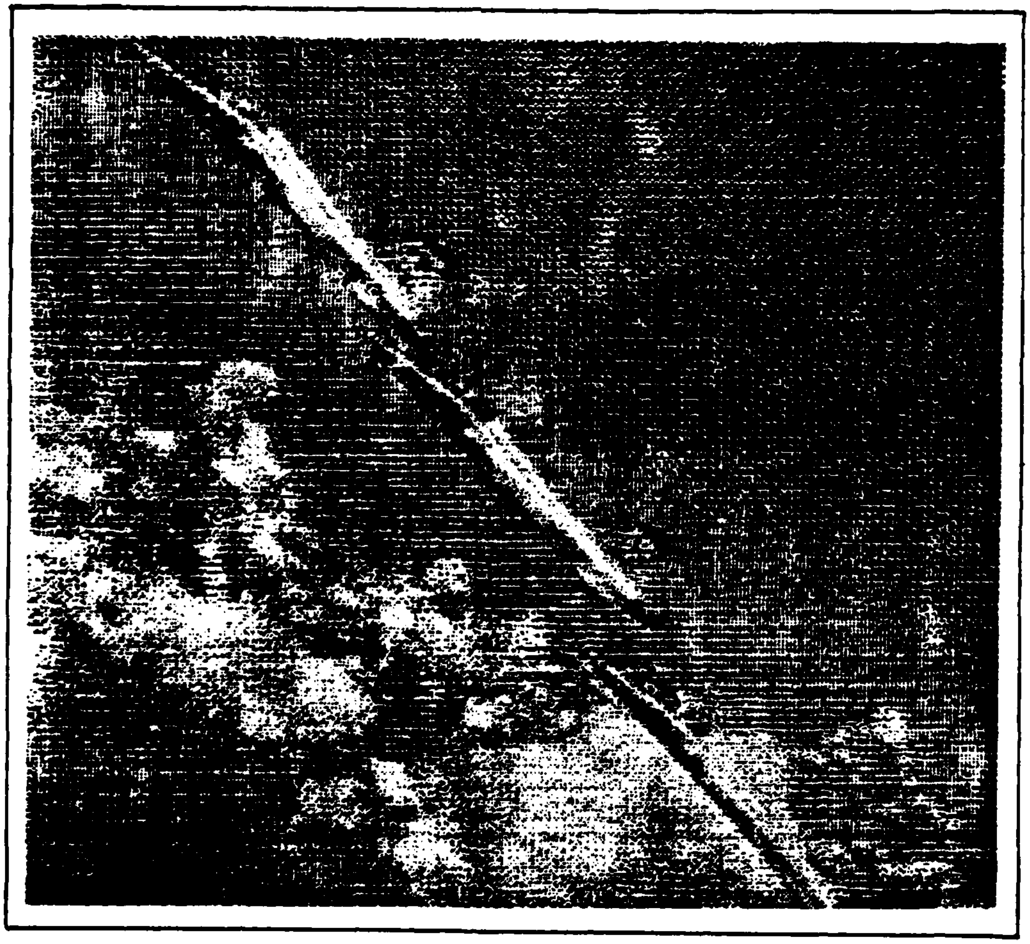


Figure 6 - Original T38 Talon

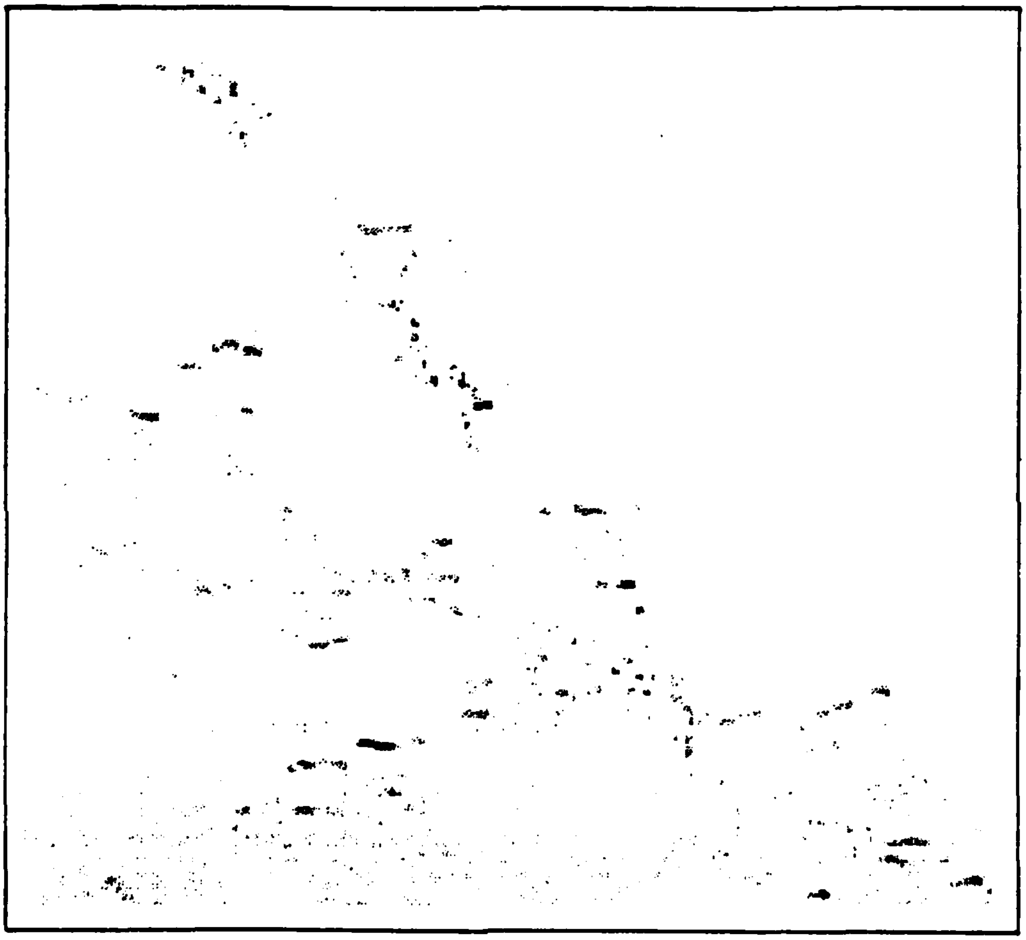


Figure 7 - Wedge Operation on T38 at Orientation #1

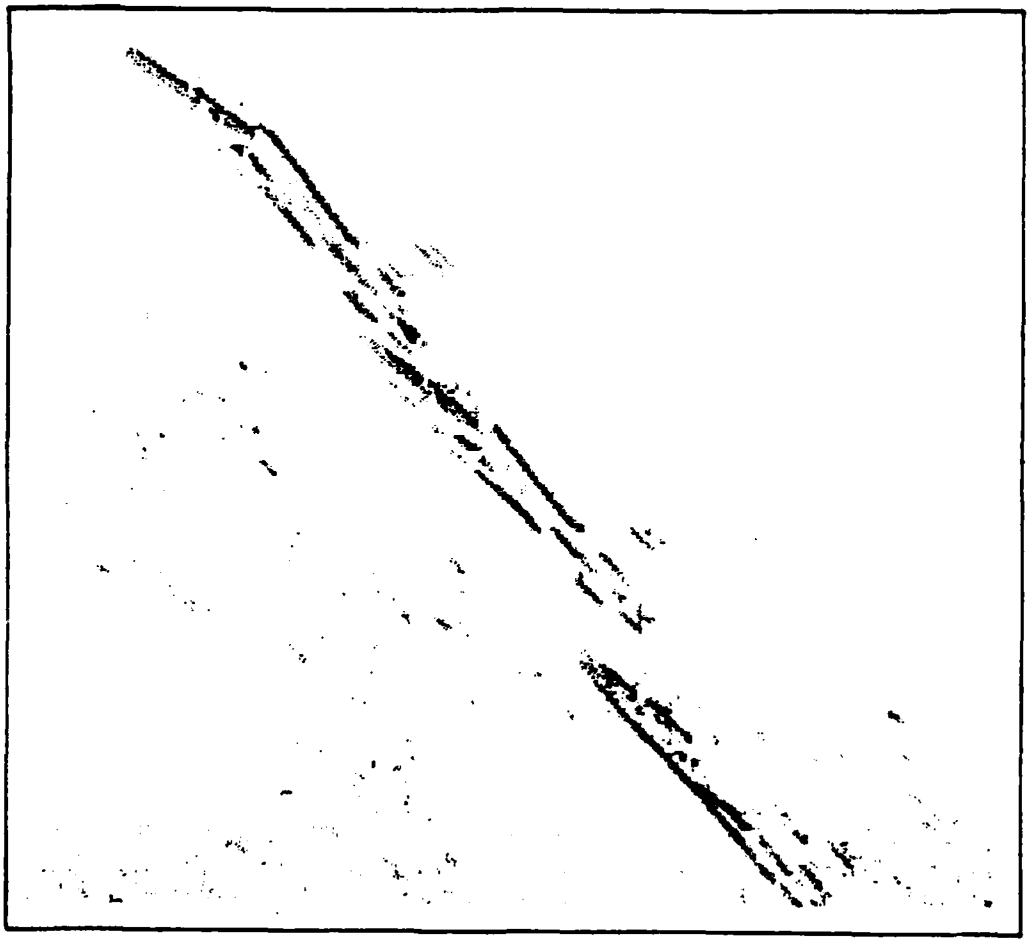


Figure 8 - Wedge Operation on T38 at Orientation #2

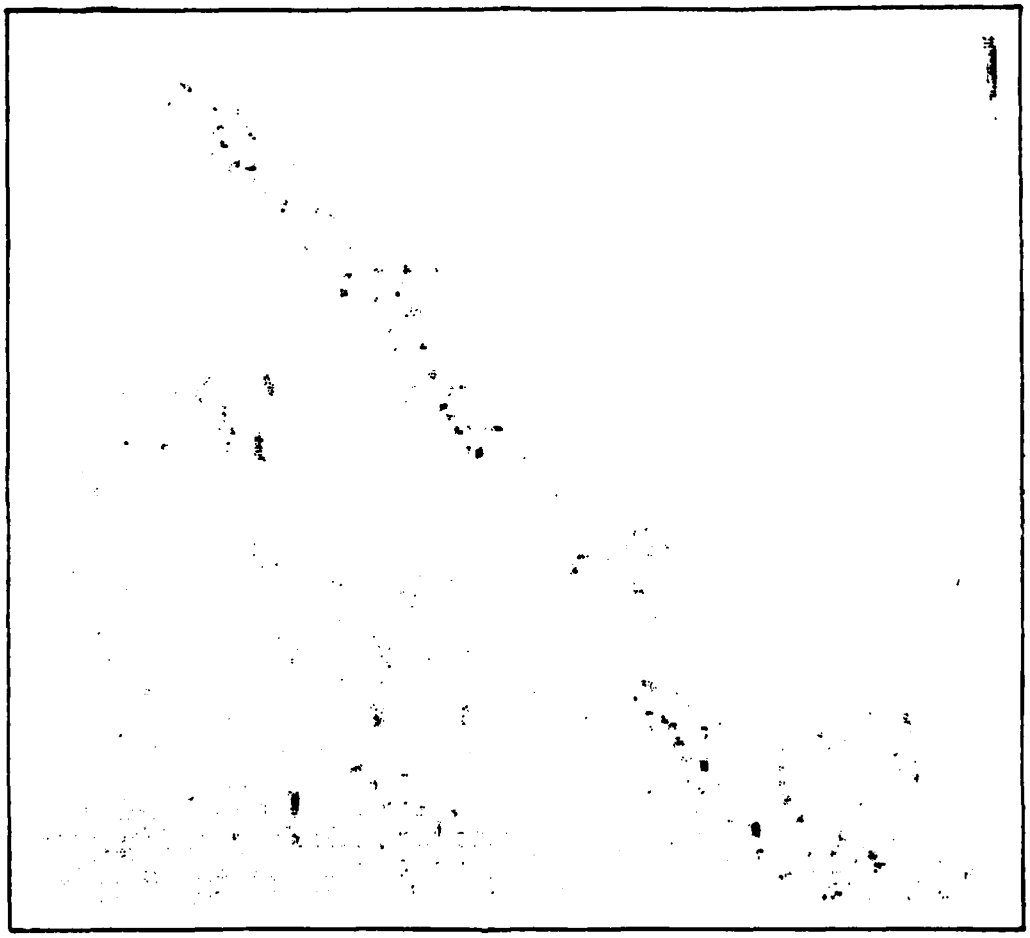


Figure 9 - Wedge Operation on T38 at Orientation #3

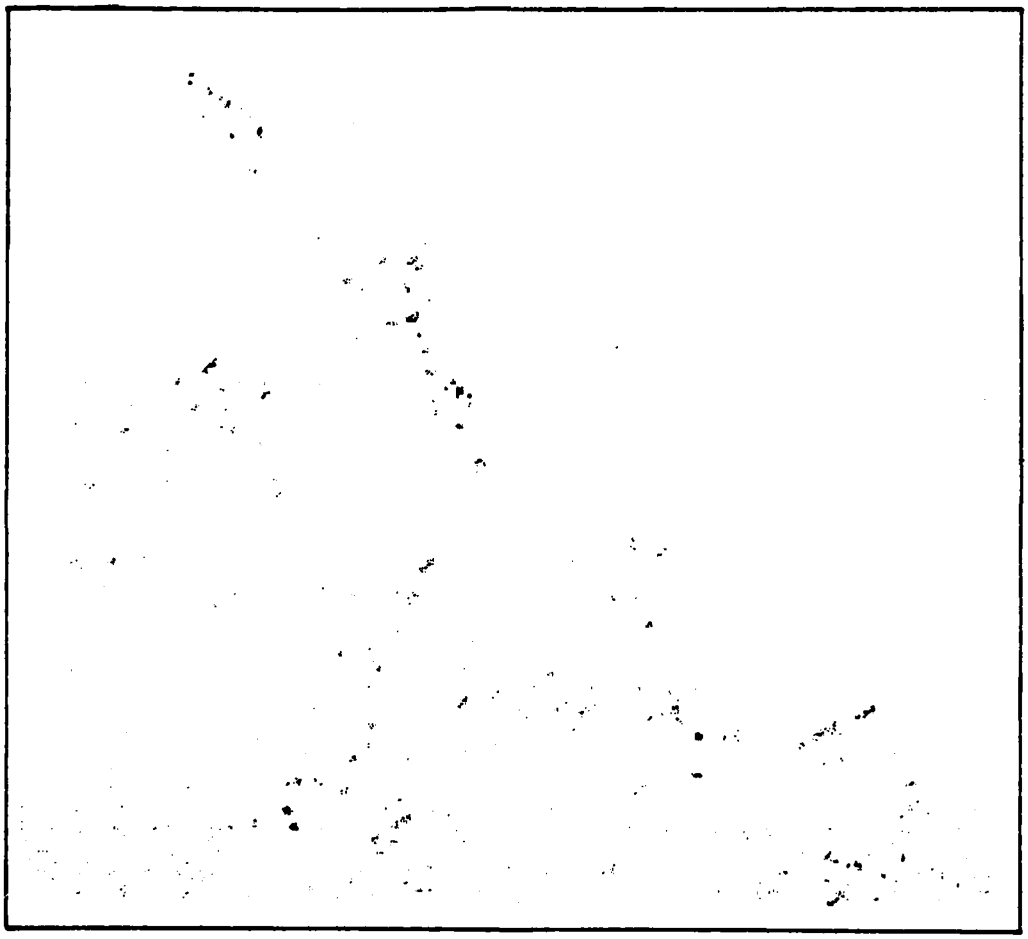


Figure 10 - Wedge Operation on T38 at Orientation #4

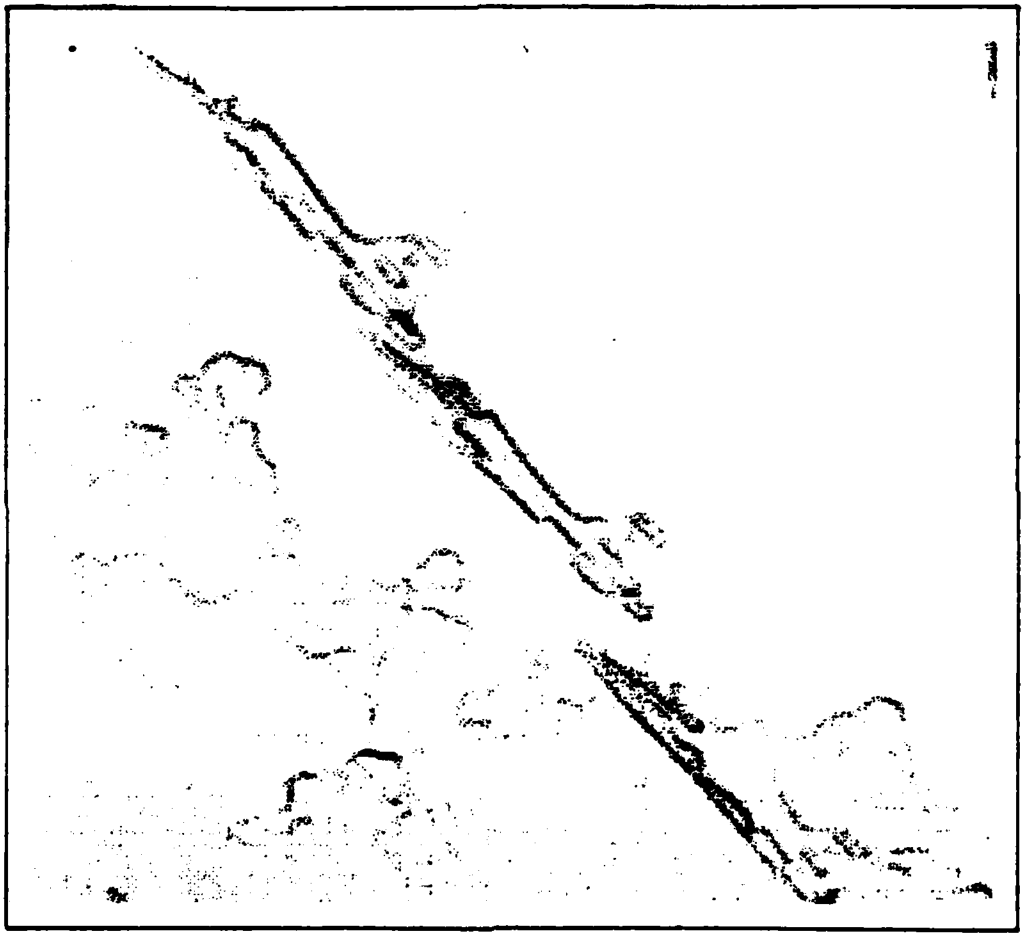


Figure 11 - Combined Result of Wedge Operator on T38

III. TEMPLATE MATCHING WITH EDGES

MATCHING

To prove the utility of edged images in template matching, real scenes were searched using an existing cross-correlation scheme (Ref 8:480). The template is in the form of an edge image of the object of interest which is to be found in the searched edged image. The template version of the object must have the same size, perspective, and orientation of the object embedded in the searched scene. The cross-correlation scheme will only correct for translation along the cartesian coordinates.

The cross-correlation scheme operates in the spatial frequency domain. The two-dimensional fast Fourier transforms (2DFFT) of the searched image and the template image are formed. The 2DFFT's represent the spatial frequency content of the images. Then the 2DFFT's are point by point multiplied to form the 2DFFT of the correlation image, which is inverse transformed to the spatial version of the correlation image. The programs to form the 2DFFT, inverse the 2DFFT, and perform complex conjugate multiplication are DIRECT, INVERSE, and CMULT. Descriptions of DIRECT and INVERSE, and the listing of CMULT are in Appendix A.

Since DIRECT, INVERSE, and CMULT require the images in complex format, programs are needed to convert between packed video format (4 bit integer) and complex format. VDTOCP converts from packed video to complex. CPTOVD converts from complex to packed video. In addition, CPTOVD will also locate the maximum value of the image to be converted. The maximum will be of interest when the image is

a correlation image. The listings of VDTOCP and CPTOVD are in Appendix A.

The maximum value in the correlation image indicates the row and column translation of the template image required to register with the most closely matching part of the searched image. Since the 2DFFT assumes periodicity of the image, translation causes wrap-around of end points. That is, points moved down past the bottom boundary go to the top of the image, and points moved past the right boundary go to the left boundary of the image.

More important than the maximum value, the peaks in the correlation image identify areas that are close to the template in match. The shape of the peaks is important, too. Sharp peaks indicate a sure match. Dull peaks indicate a close but unsure match.

TEMPLATE GENERATION

The template image is composed of lines at the edges of the object of interest. The template is zero everywhere else. The lines are assigned a maximum value of fifteen. A variable weighting method can be used to represent the importance of an edge in the matching process, but is beyond the scope of this thesis. The generation of the template image is not a trivial task.

In order to generate a template with the same scale, orientation, and perspective as the object in the searched image, the template must be digitized at the same time the searched picture is digitized. The templates were digitized from a transparency film with a clean white sheet of paper behind it. The procedure for digitizing the searched scene and the template follows:

1) Tape the transparency to the picture near the bottom so that it can be folded down out of the way when digitizing the picture.

2) Trace with a fine tipped transparency marker the edges of the object of interest.

3) Set up the picture with transparency taped to it in front of the digitizing camera and adjust camera.

4) Digitize the picture with the transparency folded down out of the way.

5) Without moving anything except for flipping the transparency back up in front of the picture, (the trace should still be aligned with the picture) digitize the transparency with a clean sheet of white paper behind it.

The digitized template has to be cleaned up because the lines will not have values of fifteen and the background will not be zero.

The program TONER, listed in Appendix A, was used to map pixel values less than eight to fifteen, and to map pixel values greater than or equal to eight to zero. This mapping negates the digitized version. Negation is necessary because the digitizer assigns a value of fifteen for full brightness, and the lines on the transparency are dark.

The template image after clean-up should be a line trace version of the object with the same orientation, size, and location of the object. The template can then be moved with a known translation. The program NMOVE, listed in Appendix A, was used to translate the template to the center of the image. Figure 12 shows the finished template created for the image shown in Figure 6.

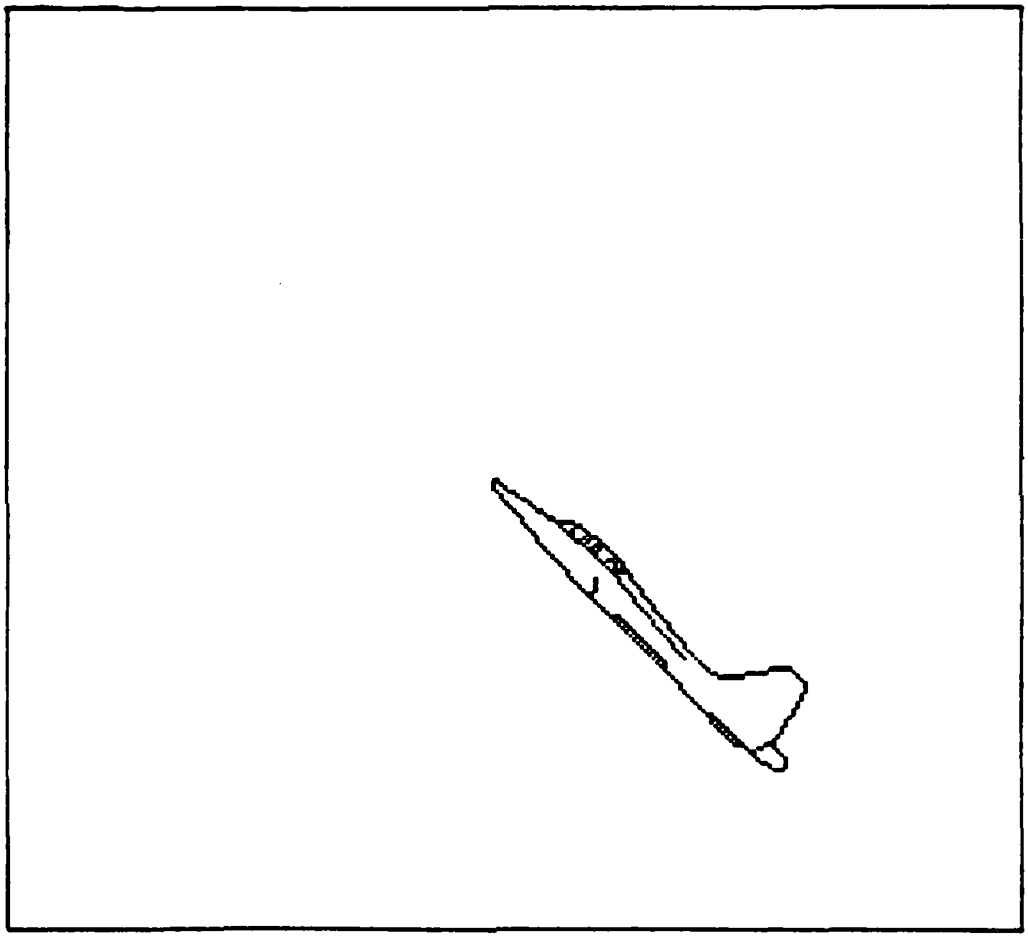


Figure 12 - T38 Template

RESULTS

The image shown in Figure 6 was searched using the template shown in Figure 12. The cross-correlation image is shown in Figure 13. The peaks are fairly sharp and the maximum corresponds to a template translation that is one pixel location from the correct translation.

Figure 6 is a good test for the discrimination ability of the matching process since there are several other objects of interest at approximately the same size, orientation, and perspective. The other peaks in Figure 13 that do not contain the maximum correspond to the other objects of interest in the searched image. The other peaks have maximum values that are 90% or better of the global maximum.

Several other pictures were searched similarly and the targets were located within one or two pixel locations. These results are in Appendix C.

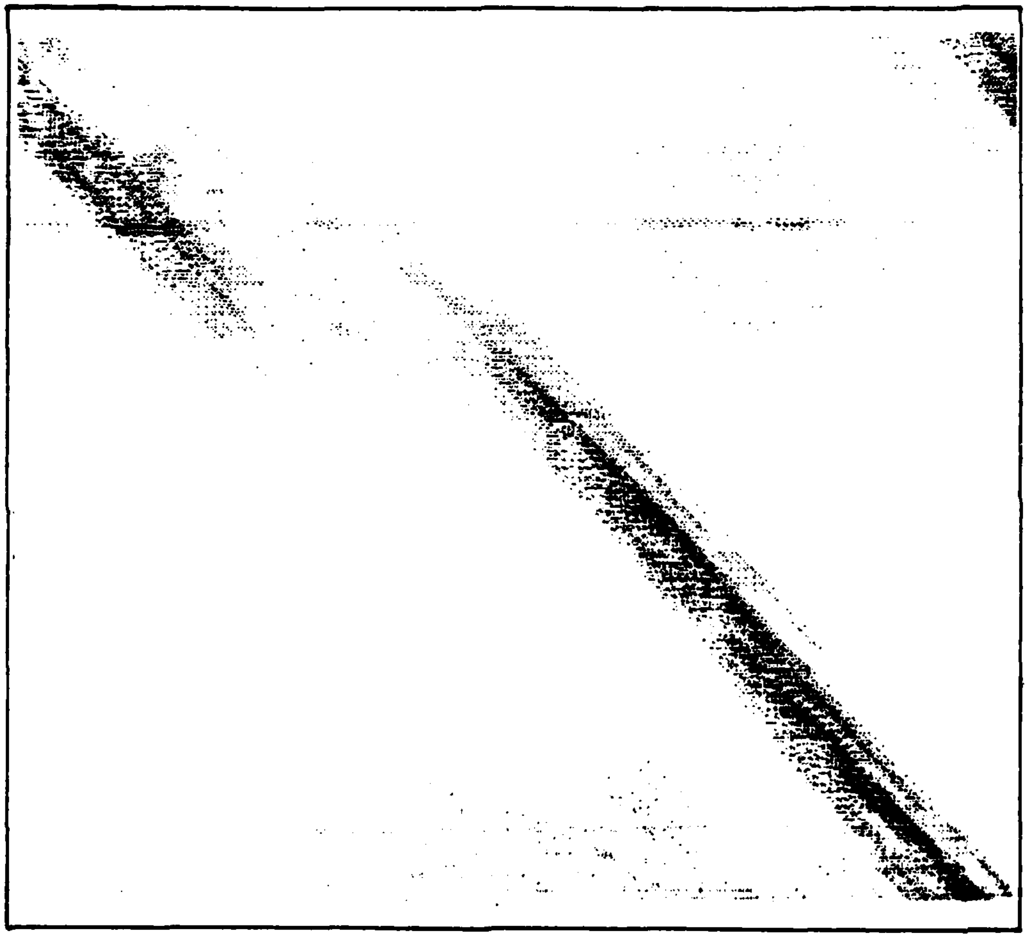


Figure 13 - T38 Correlation Image

SEPARATED ORIENTATION CORRELATION

With hopes of improving the correlation scheme with edges, the separated edge orientation method was combined with the cross-correlation method. The method of separating edge output into four different images with the wedge operator is described by the section on mask edging in Chapter II. We want to correlate the edge image at orientation #1 of the searched image with the edges at orientation #1 of the template. Similarly, we want to correlate the edges at orientations #2, #3, and #4. Then, add the correlation output for each of the orientations to form the final correlation image.

To separate the template image into edges at four different orientations, a line detection and separation operator was devised. The tedge operator, as hereby named, is very similar to the wedge operator in that it uses convolution masks. Each of the four masks shown in Figure 14 is convolved with the neighborhood of the pixel being processed. The mask orientation that produces the maximum convolution value determines which output image the pixel will be assigned to. If two or more masks produce the same maximum, the edge pixel will be assigned to each corresponding output image. The program for the tedge operator is TEDGE and is listed in Appendix B. Figures 15-18 show the separated edge orientations for Figure 12.

The correlation is performed as described in the previous section on matching. The edges are correlated at the separate orientations and then the correlation results are summed. The program to sum the correlation results when they are in complex format is ADDCMP as listed in Appendix A. Figure 19 is the result of performing the separated edge orientation correlation of Figures

7-10 with Figures 15-18 respectively. Notice that peaks are sharper. Also, the target was located exactly by the maximum correlation value.

Orientation #1:

0	0	0
1	1	1
0	0	0

Orientation #2:

1	0	0
0	1	0
0	0	1

Orientation #3:

0	1	0
0	1	0
0	1	0

Orientation #4:

0	0	1
0	1	0
1	0	0

Figure 14 - Convolution Masks for Tedge Operator

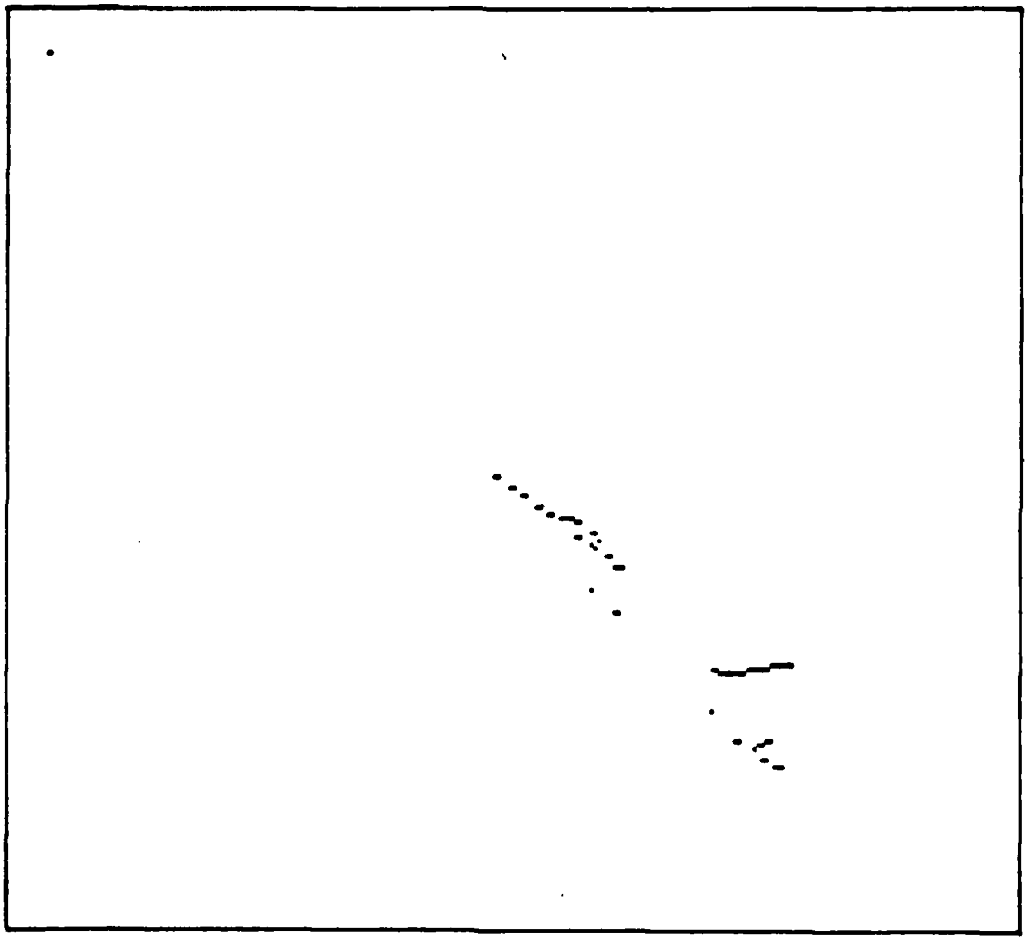


Figure 15 - Tedge Operation on T38 Template at Orientation #1

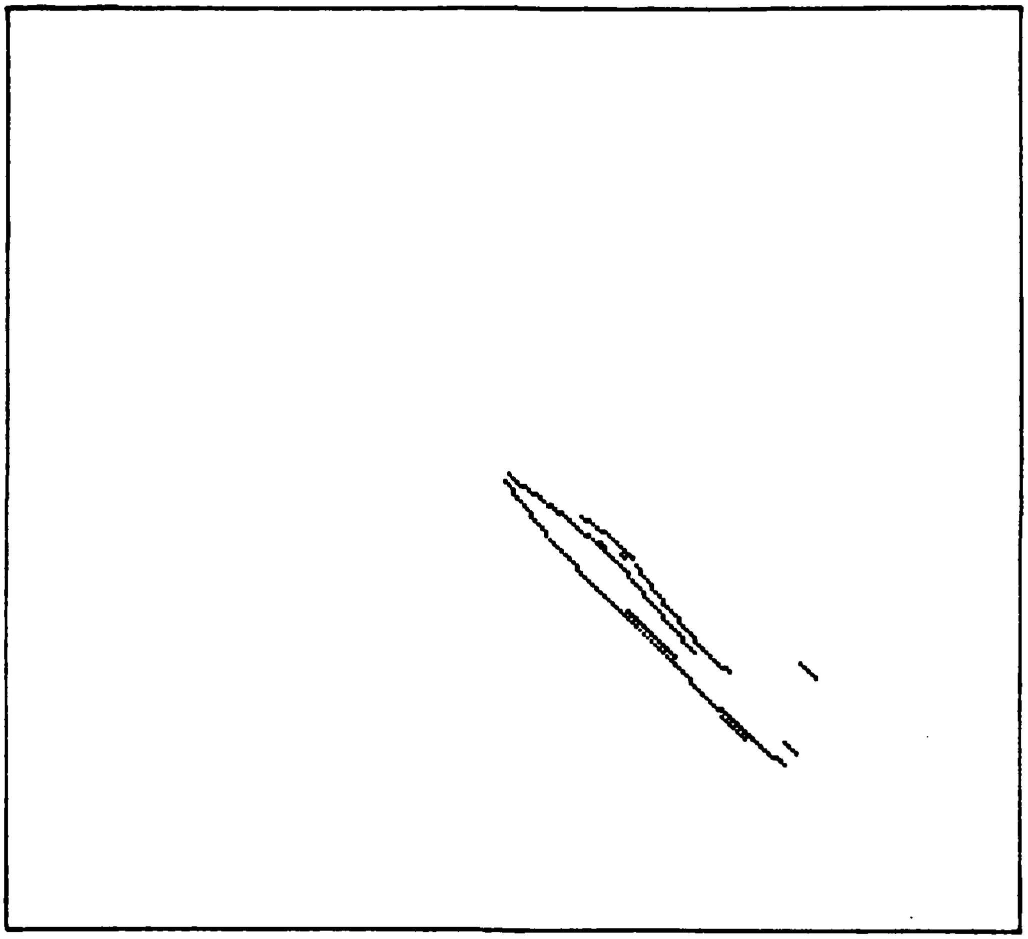


Figure 16 - Tedge Operation on T38 Template at Orientation #2

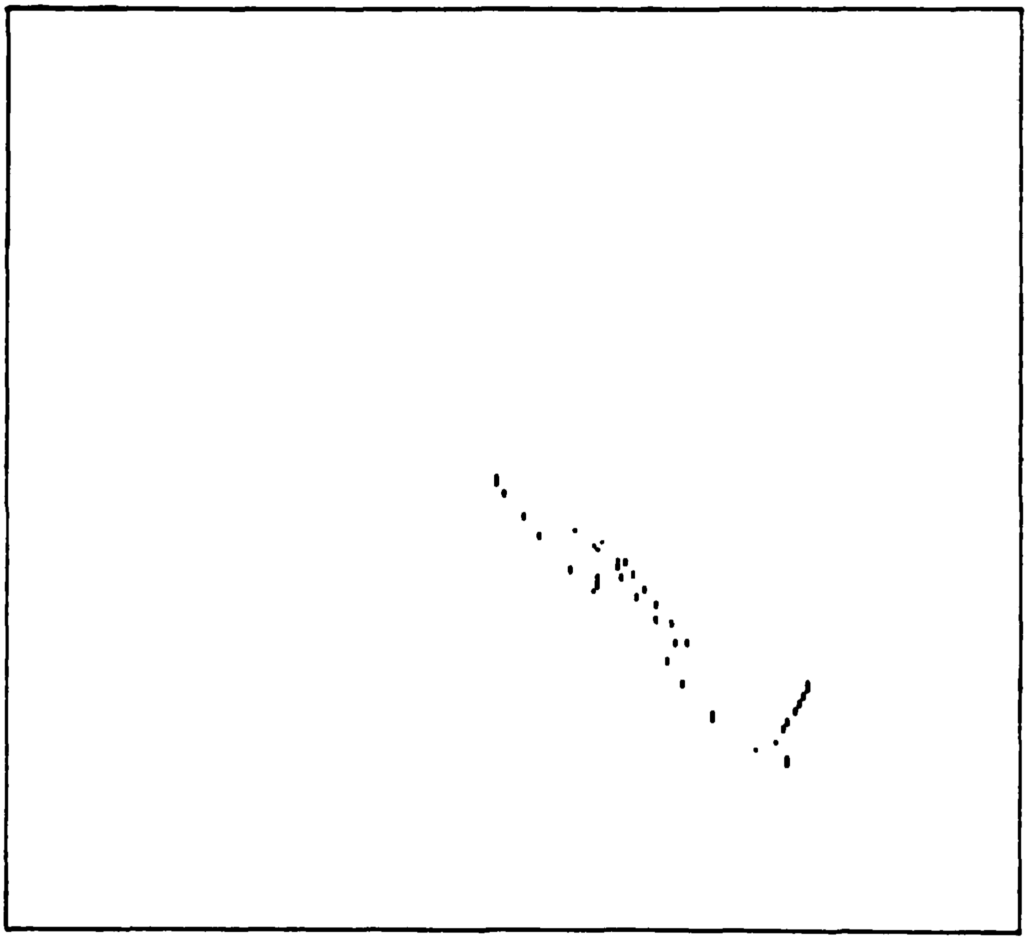


Figure 17 - Tedge Operation on T38 Template at Orientation #3

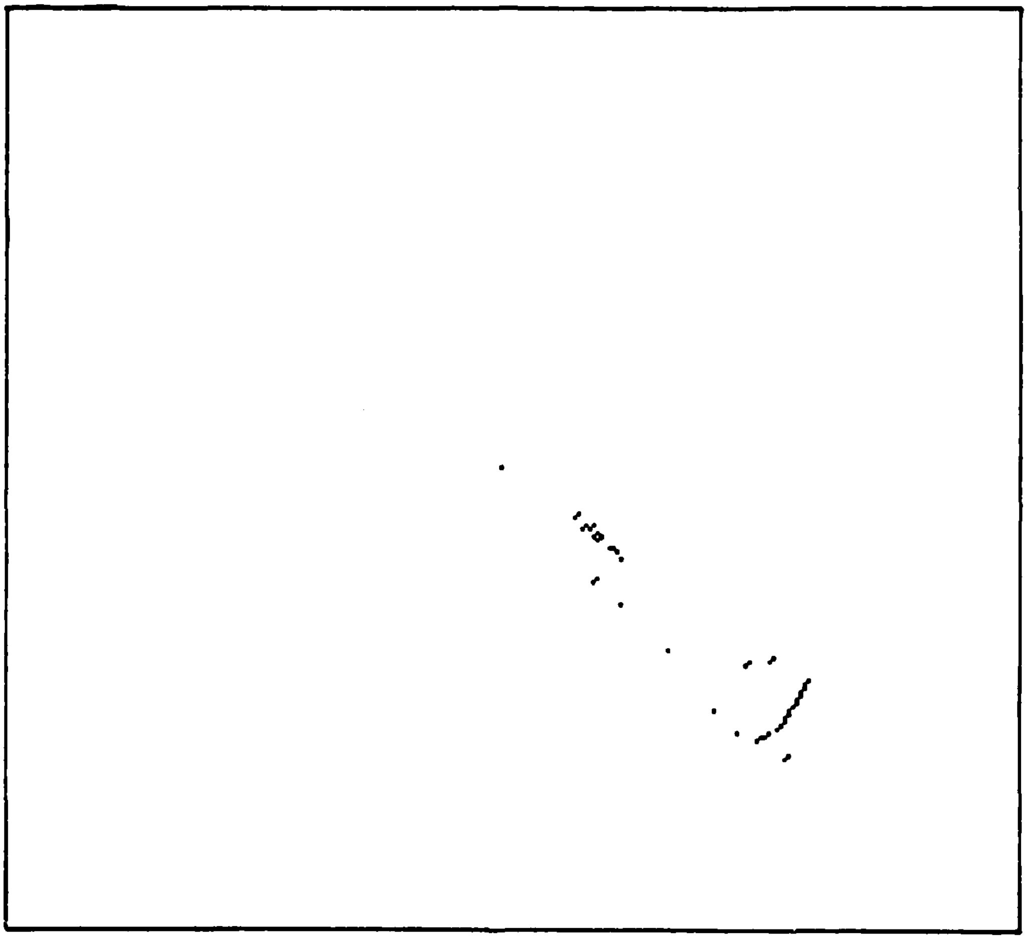


Figure 18 - Tedge Operation on T38 Template at Orientation #4

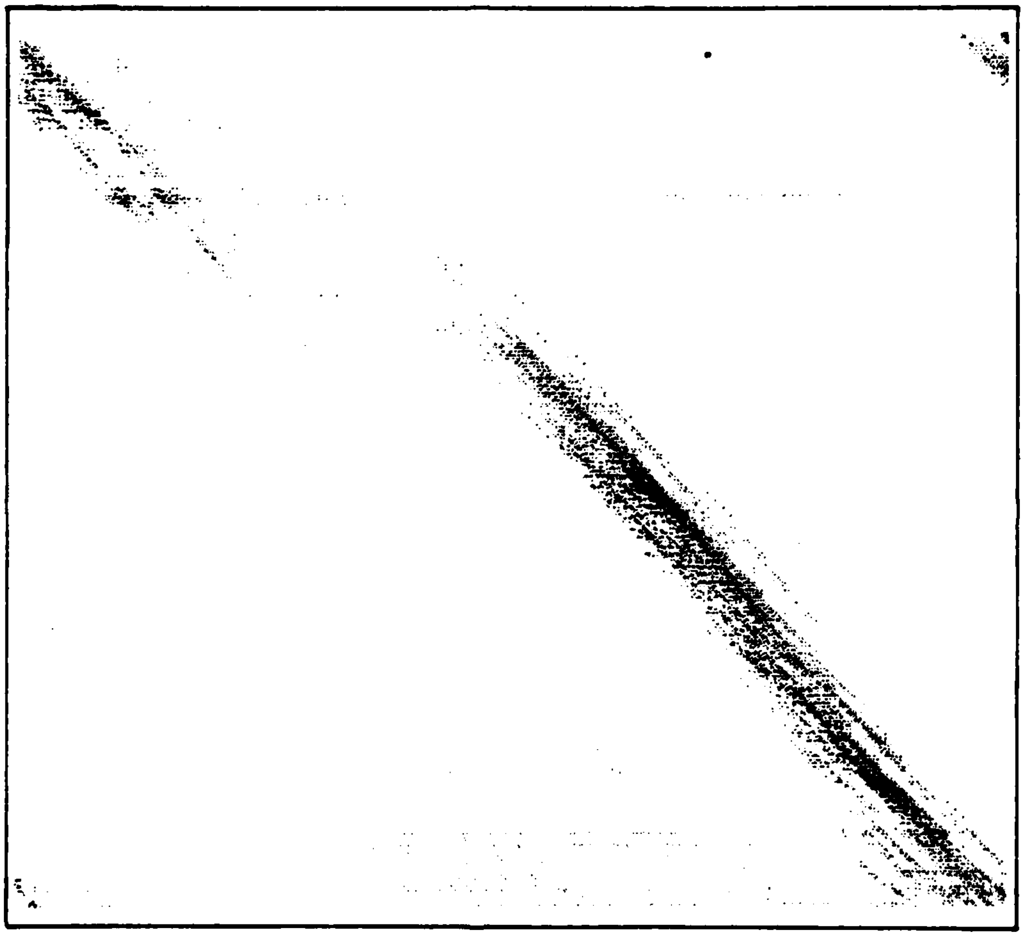


Figure 19 - T38 Separated Orientation Correlation Image

IV. CLUSTER RECOGNITION

HAMADANI ALGORITHM

In 1981, N.A. Hamadani developed an automatic target cuer for infrared images. The algorithm is basically a cluster recognition process that operates solely in the spatial domain. The clusters in the infrared images are "hot" blobs that typically represent man-made objects with much thermal activity, such as trucks or tanks. Hamadani's algorithm was implemented here in the Signal Processing Laboratory with the intention of researching improved preprocessing techniques.

The algorithm has seven basic steps as described below. The first three steps are considered preprocessing. Only the first four steps were actually implemented with the assumption that, given the understanding of the remaining steps, the results of the entire algorithm may be known. The steps are:

1) Enhance the original image with a convolutional mask operator that has the same pattern as the Kirsch operator, but the coefficients are 10 and -1 instead of 5 and -3 respectively.

2) Edge extract the enhanced image with the Kirsch operator.

3) Threshold the enhanced image using a global conjunctive thresholding method. Global statistics form the thresholds for the enhanced and edged images. Where both the enhanced image is above its threshold and the edged image is above its threshold, a value of fifteen is assigned to the output. Otherwise, the output at that pixel location is assigned a value of zero. Conjunction refers to the thresholding on both the enhanced and edged images.

The global threshold for each is its global mean plus its global standard deviation.

4) Reduce the thresholded image using a connectedness test that states a non-zero pixel will be removed if it does not have at least two non-zero neighbors above, below, to the left, or to the right. Diagonal neighbors are ignored. The test is repeated until no pixels are removed.

5) Segment the non-zero pixels into target clusters by finding which ones are connected and assigning them a target identification number to keep track of which pixels are in a particular cluster.

6) Erode the outside boundaries of the target clusters with what Hamadani calls thinning. For thinning, the desired target shape is assumed to be rectangular. The top row of each cluster is removed if the one below it has more non-zero pixels in it. Similarly, leftmost and rightmost columns and the bottom row are tested.

7) The target is tested for acceptable size. The clusters must be at least 3 X 3 pixels, and the ratio of length of width must be between 8:1 and 1:8. Also, the number of non-zero pixels in the (assumed) rectangular boundary must be at least 75% of the maximum possible.

The program to implement the first four steps is HAM1 as listed in Appendix D. The results for the infrared image in Figure 20 after each of the four steps are shown in Figures 21-24. If the last three steps operated on Figure 24, the result would look like Figure 25, which was altered manually for inclusion here. There are indeed three targets as indicated.

Figure 20 - Infrared Image #1 (IMAG1.IR)

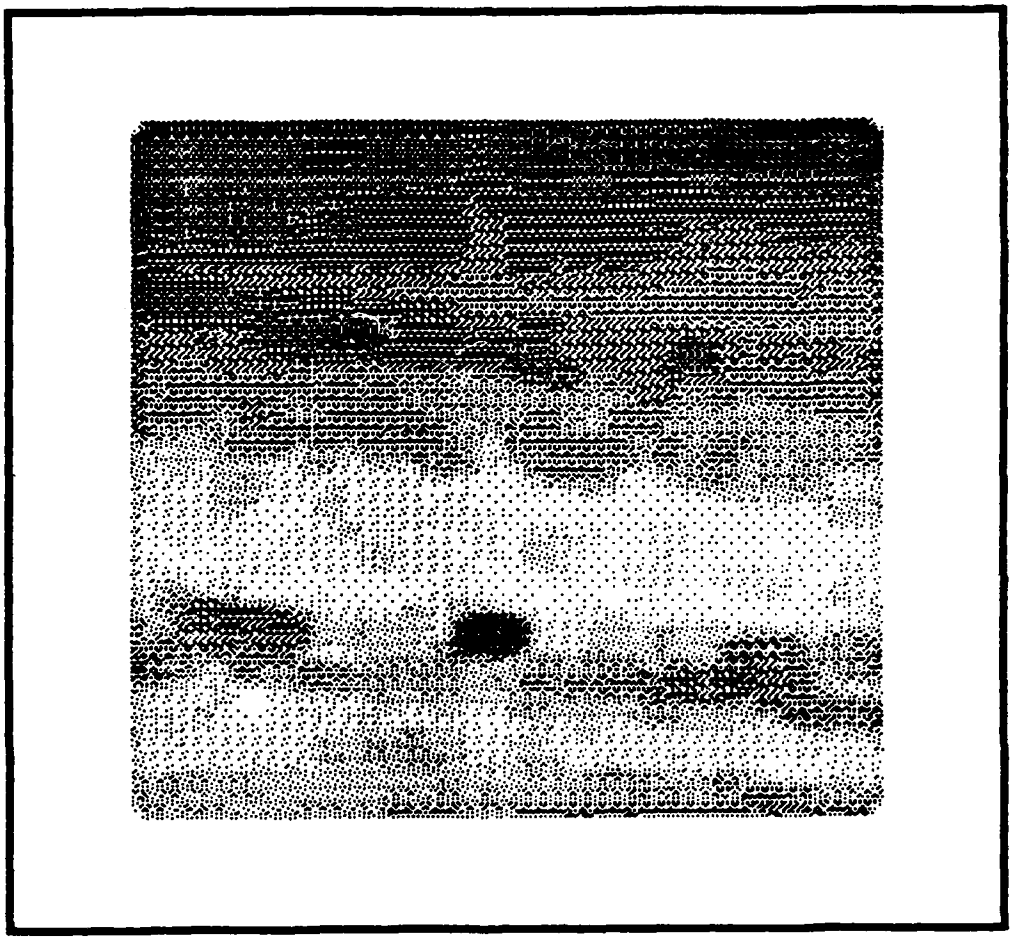


Figure 21 - Enhanced Image of IMAG1.IR

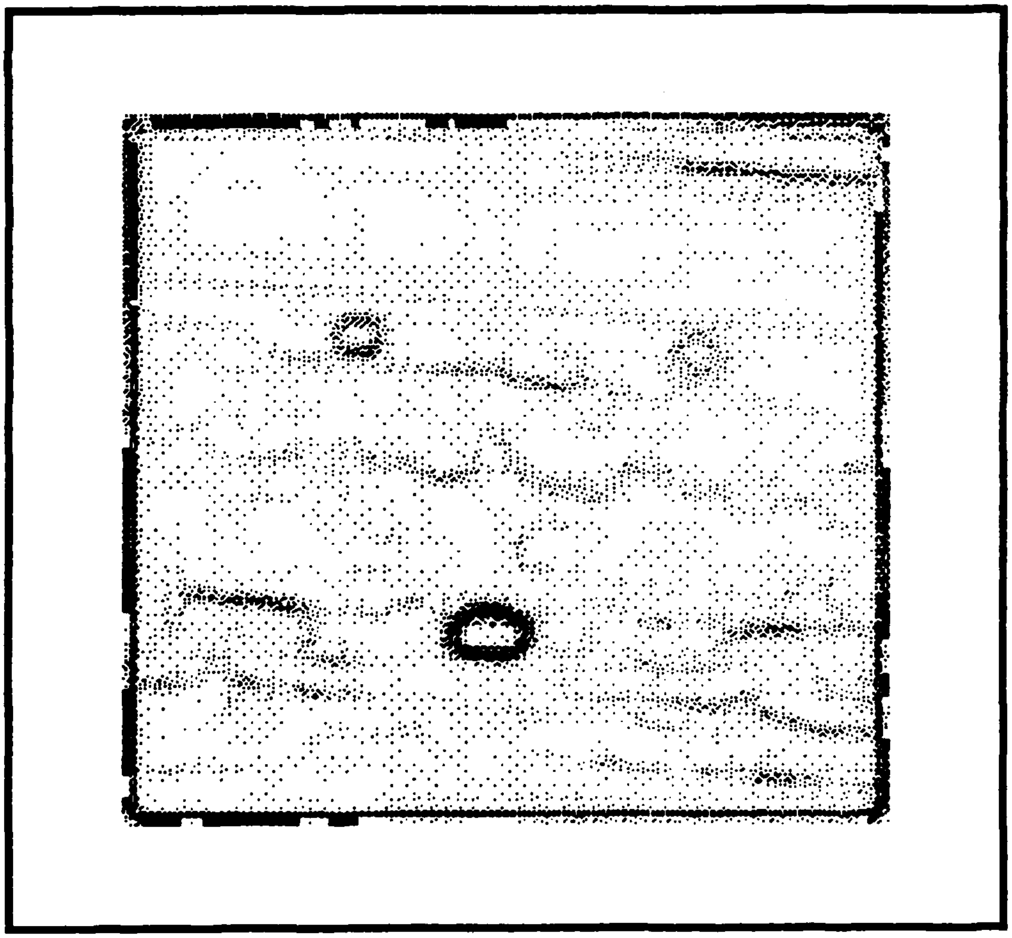


Figure 22 - Edged Image of IMAG1.IR

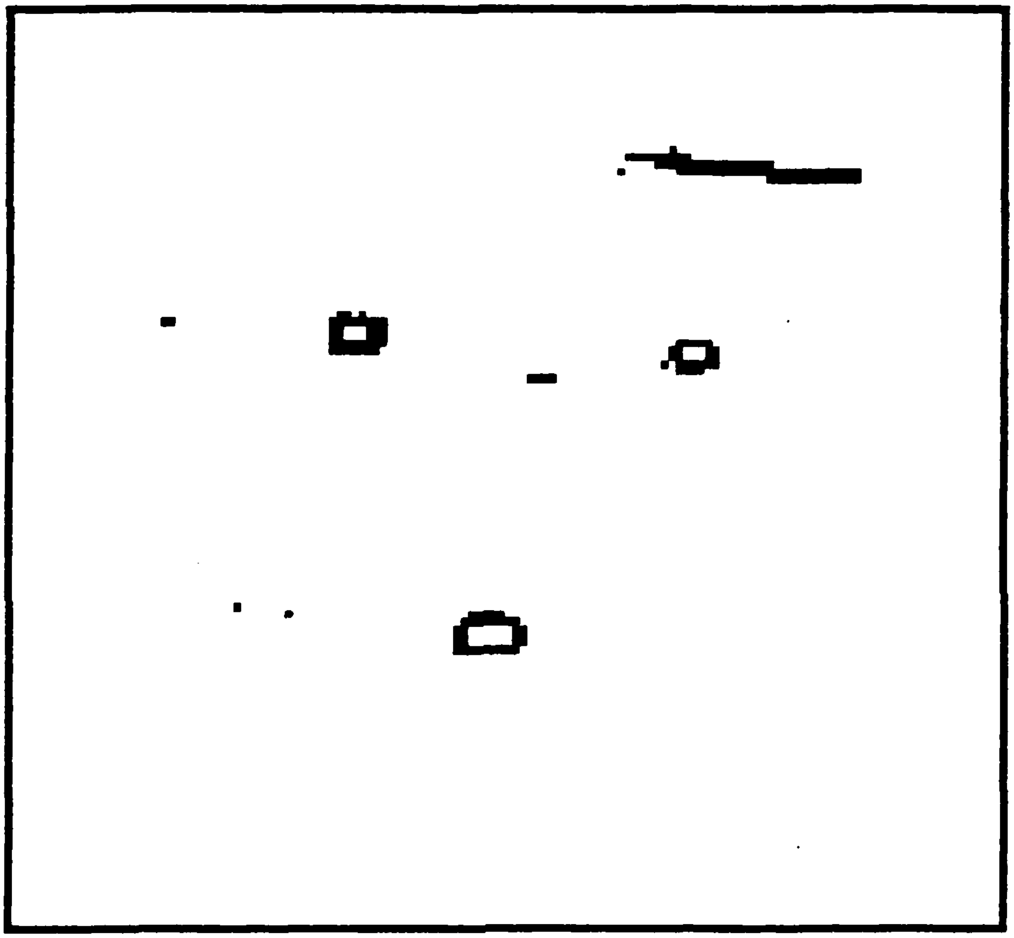


Figure 23 - Thresholded Image of IMAG1.IR

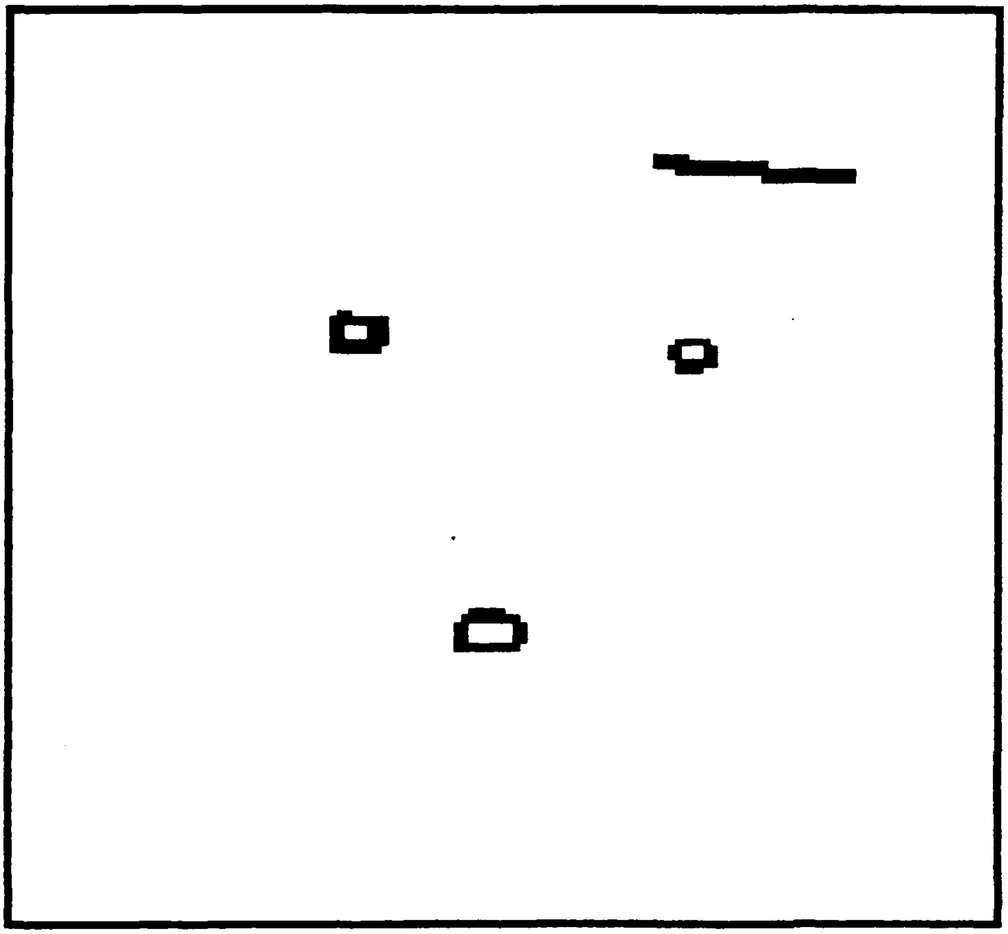


Figure 24 - Thresholded IMAG1.IR after Connectivity Test

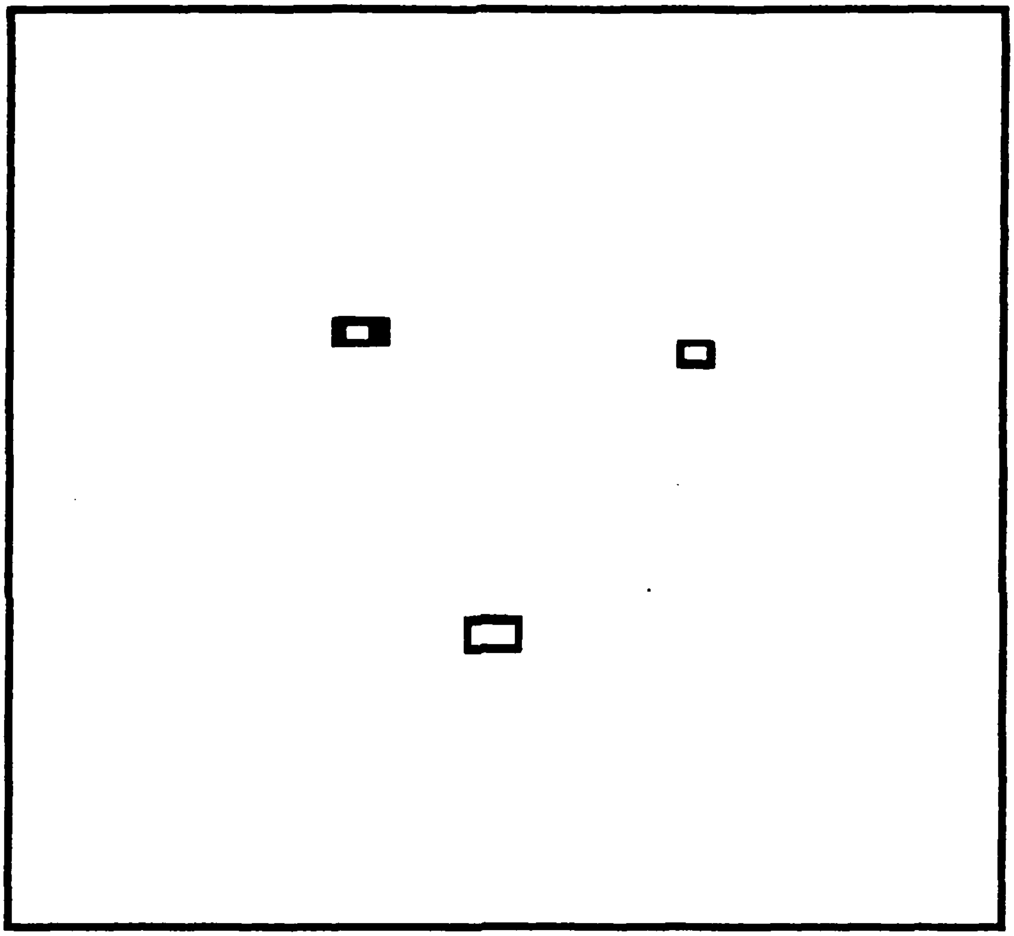


Figure 25 - Finished Image of IMAG1.IR

Variations on the edging algorithm were tested but the results were the same as with the Kirsch operator. Instead of pursuing work on the edging process, other parts of the preprocessing were changed with hopes improving the entire algorithm.

LOCAL THRESHOLDING

With all of the other steps unchanged, the conjunctive thresholding was implemented using local neighborhood statistics. The threshold for each pixel is separately computed using the mean plus the standard deviation of the square local neighborhood. The computation time increases significantly, but the process now has the ability to correct for variations in the average intensity across the image.

The results for the local thresholding were not very good until another step was also changed. When using local thresholding, the edged image must be formed from the original image instead of the enhanced image. This is because the enhanced image dilates the hot blobs, and when edged, the conjunctive thresholding shows that the thresholded enhanced and edged images will not coincide very well. By edging the original image, the edges of the hot blobs are closer to the center of the cluster. Then the thresholded edges coincide better with the thresholded enhanced image.

The program to perform the new algorithm with local thresholding is HAM21 as listed in Appendix D. The result for this version of the algorithm using a 17 X 17 neighborhood for Figure 20 is shown in Figure 26. The last three steps of Hamadani's algorithm would reduce the image to Figure 27. The three targets are again detected.

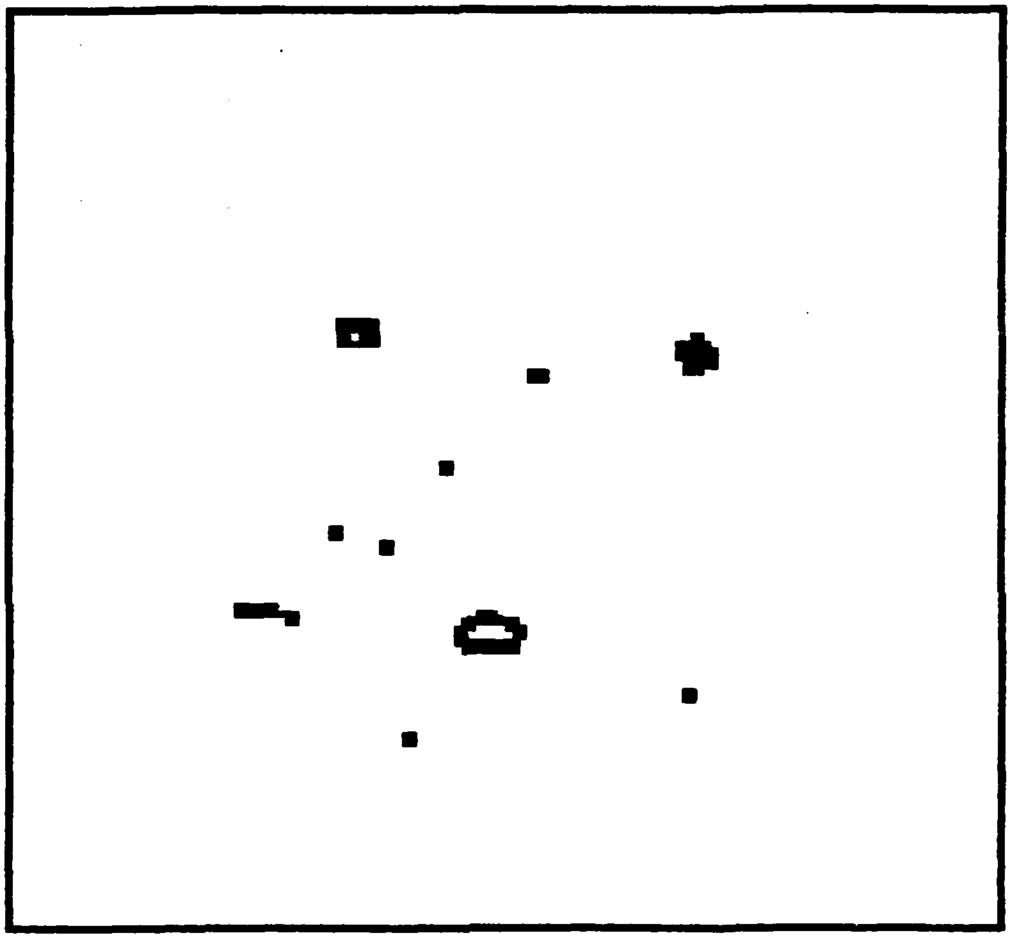


Figure 26 - Locally Thresholded Image of IMAG1.IR (Using 17 X 17 Neighborhoods) after Connectivity Test

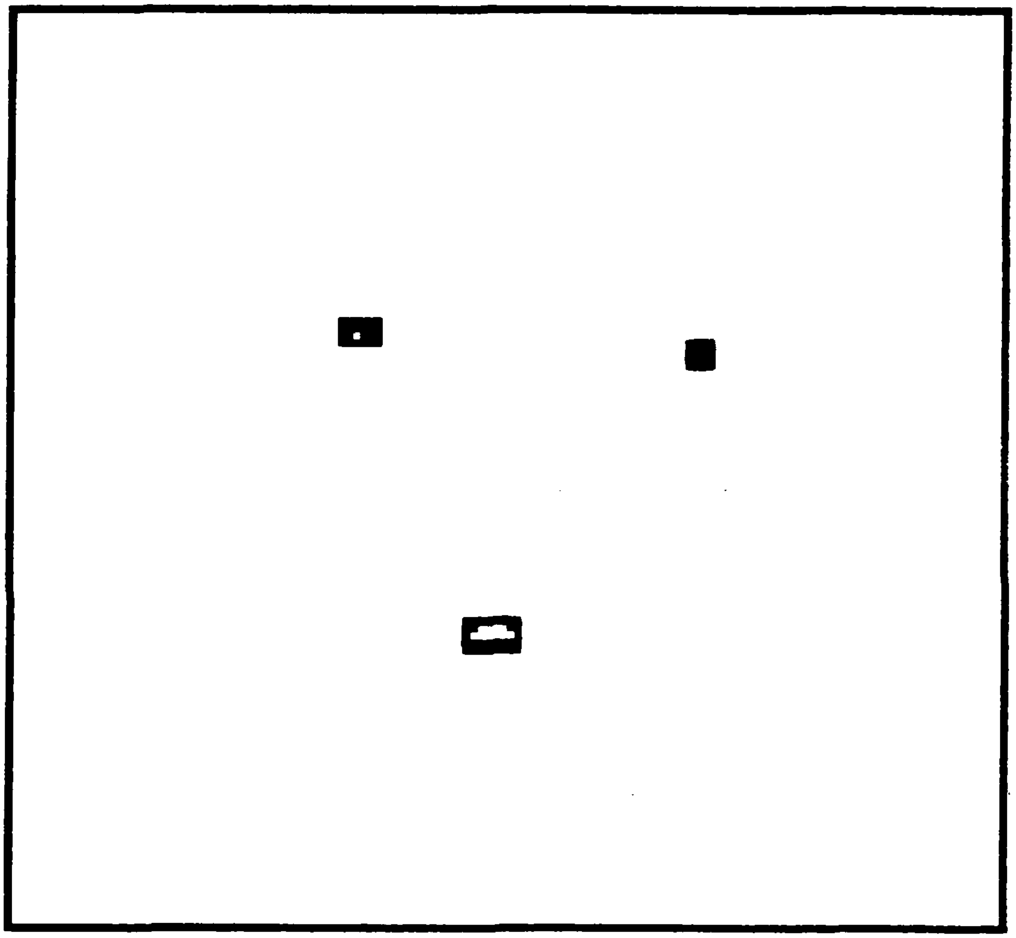


Figure 27 - Finished Image of IMAG1.IR Using Local Thresholding with 17 X 17 Neighborhood

NEIGHBORHOOD SIZE

The amount of computation for local thresholds is proportional to the square of the size of the neighborhood. We are thus motivated to try to obtain good results with smaller neighborhoods. 17 X 17 neighborhoods produce good results, but 7 X 7 neighborhoods do not unless some changes are made in the way the thresholds are computed.

Specifically, the amount of local standard deviation that is added to the local mean must be modified. A coefficient on the standard deviation was experimented with, but good results could not be produced without limiting the range of the product of the local standard deviation and its coefficient. The limits must be imposed because when the small neighborhood is placed over a high contrast edge, the standard deviation will be unappropriately high. Thus, a maximum must be found.

In addition, small neighborhoods allow some false targets to appear in the results. This is because in relatively edgeless areas the standard deviation is inappropriately low. Thus, a minimum must be found. When the standard deviation is modified by a coefficient, a minimum, and a maximum, the results for local thresholding with 7 X 7 neighborhoods are better than ever.

The program for local thresholding with the modified standard deviation term is HAM31 as listed in Appendix D. The result of using a 7 X 7 neighborhood in HAM31 on Figure 20 is shown in Figure 28. The last three steps of Hamadani's algorithm would reduce the image to Figure 29.

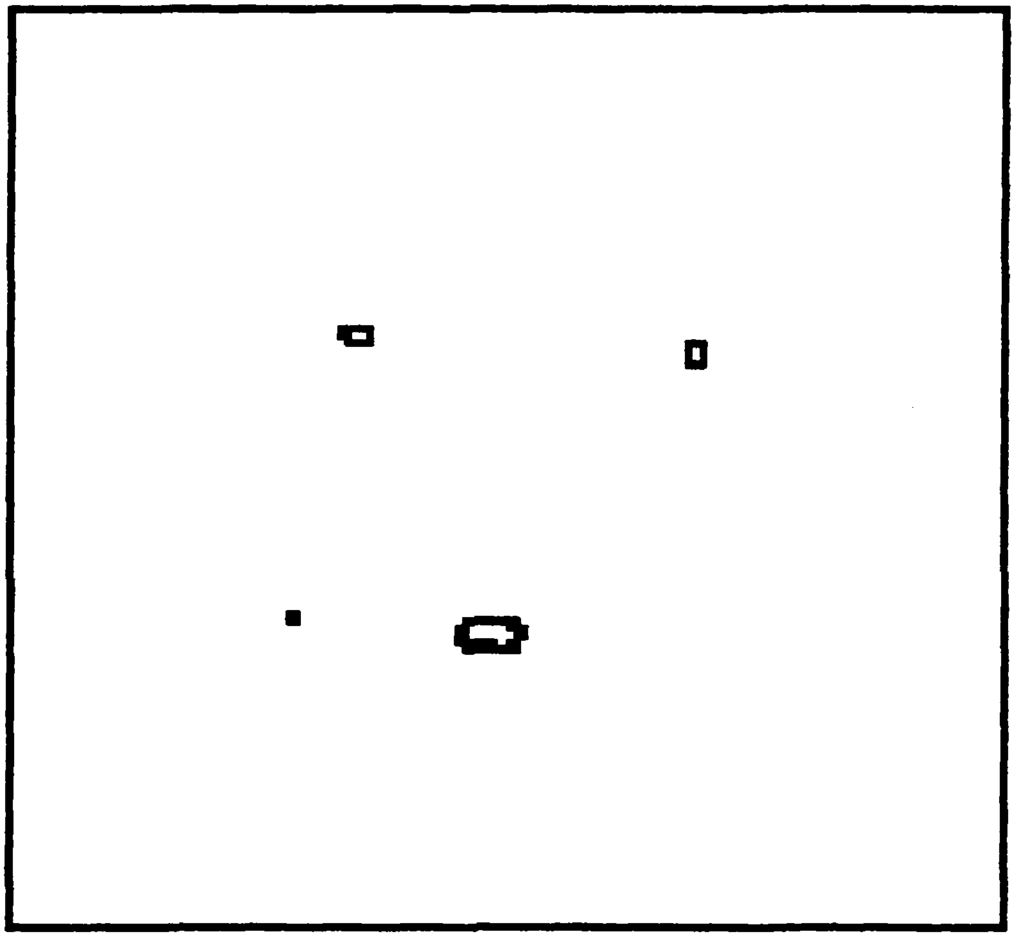


Figure 28 - Locally Thresholded Image of IMAG1.IR (Using 7 X 7 Neighborhoods) after Connectivity Test

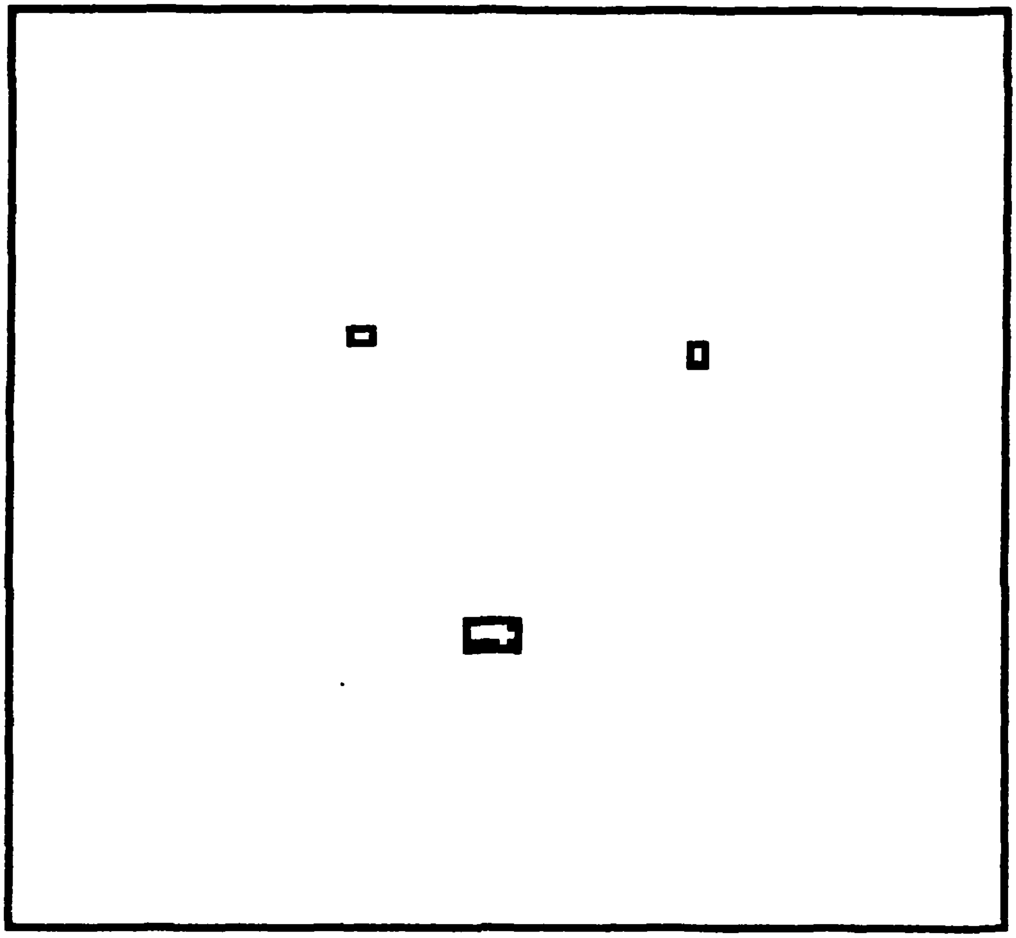


Figure 29 - Finished Image of IMAG1.IR Using Local Thresholding with 7 X 7 Neighborhoods

THRESHOLD RULE

While making the previously mentioned improvements in Hamadani's algorithm, the results of thresholding the enhanced image were viewed separately from the thresholded edge image. This allows inspection of what exactly the conjunctive thresholding does. With local thresholding, the process was observed to depend mostly on the thresholding of the enhanced image.

The edging step was then omitted to test this observation for another shortcut. The thresholding is now only on the enhanced image. The program to implement this shortcut is HAM61 as listed in Appendix D. The result of using a 7 X 7 neighborhood in HAM61 on Figure 20 is shown in Figure 30. The last three steps of Hamadani's algorithm would reduce the image to Figure 31.

The preprocessing using Hamadani's first four steps and all of the described modifications were tested on eight different images. The results are included in Appendix E.

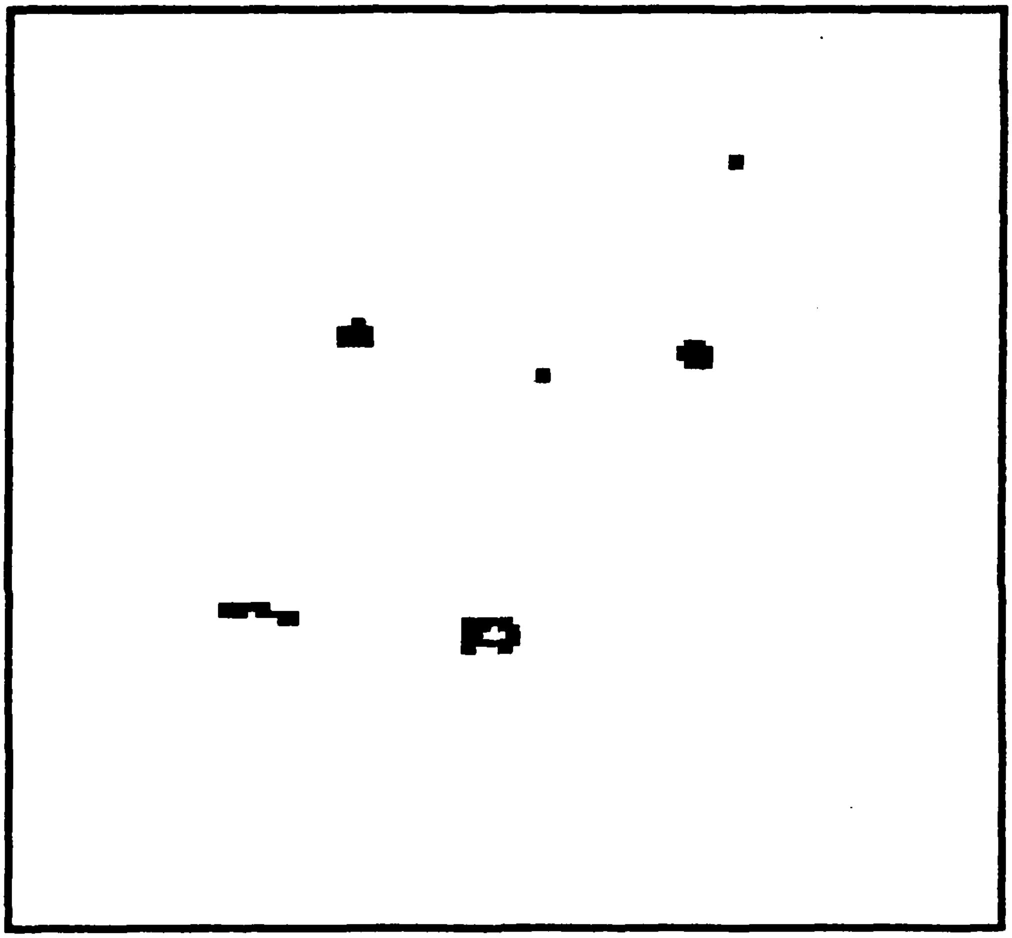


Figure 30 - Locally Thresholded Image of IMAG1.IR (Using 7 X 7 Neighborhoods and No Edging) after Connectivity Test

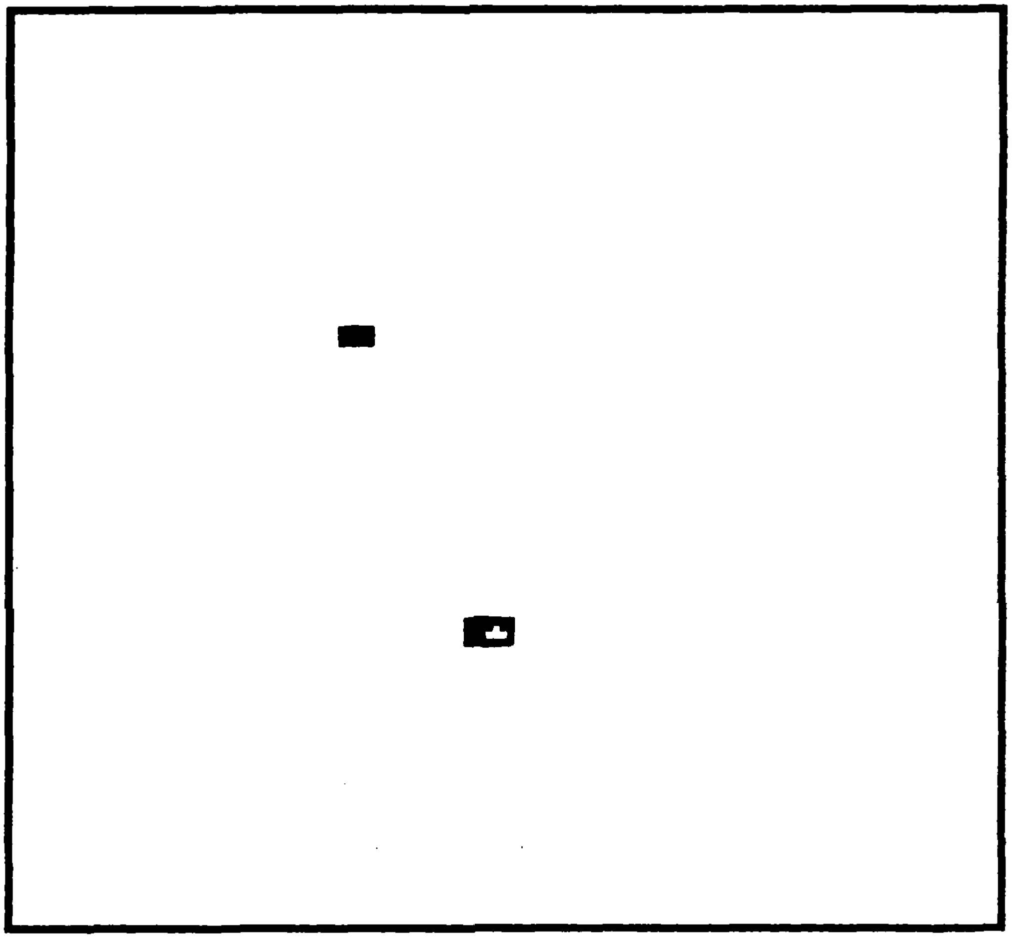


Figure 31 - Finished Image of IMAG1.IR Using Local Thresholding with 7 X 7 Neighborhoods without Edging

V. CONCLUSIONS AND RECOMMENDATIONS

EDGING OPERATORS

Large masks for edging operators only decrease the resolution of the edges but may help the overall process if the images are noisy. The best masks represent an ideal edge at various orientations. By finding the mask that produces the maximum convolution sum, edging determines the contrast and the orientation of the edges.

Mean deviation is very similar to difference, gradient, or derivative operators such as Roberts and Sobel operators for extracting edges. These are all more susceptible to noise than the edge mask approach. Mean deviation is preferred to standard deviation because it is faster to compute and is less affected by noise.

CORRELATION WITH EDGES

Template matching can be implemented well by correlation with edges. The template is a line trace drawing of the object of interest. The correlation process described in this effort corrects target translation only. When separated orientation edge images are used, the correlation process is further improved. Some research into whether more orientations help the process is suggested. Also, an orientation correction correlation scheme might be devised using the separated orientation edge images.

CLUSTER RECOGNITION

The Hamadani algorithm works well for detecting clusters in the infrared images. Local thresholding improves the process by compensating for changing average intensity across the image. 17 X 17 neighborhoods prove to work well for local thresholding. 7 X 7 neighborhoods can be used if the threshold computations are slightly modified. Less than 7 X 7 neighborhoods could not be implemented with good results. When local thresholding is used, the conjunctive threshold rule can be reduced to thresholding the enhanced image only. Thus the edge image can be omitted.

Further work on the cluster recognition algorithm might include computing the thresholds using selective neighborhoods that are perpendicular to the edges. Enhancement might be performed more than once. The last four steps of Hamadani's algorithm are subject to research. Some better connectivity tests, thinning routines, and size tests could be investigated.

BIBLIOGRAPHY

1. Rosenfeld, A. "Image Pattern Recognition," Proceedings of the IEEE, 69:5, pp. 596-605, May 1981.

2. Kabrisky, M. Lecture notes, EE6.20, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1983.

3. Hamadani, N.A. "Automatic Target Cueing in IR Imagery," MS Thesis, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December 1981. (AFIT/GEO/EE/81D-3)

4. Cromer, J. "Scene Analysis: Non-Linear Spatial Filtering for Automatic Target Detection," MS Thesis, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December 1982. (AFIT/GE/EE/82D-26)

5. Simmons, R.A. "Machine Segmentation of Unformatted Characters," MS Thesis, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December 1981. (AFIT/GE/EE/81D-54)

6. Feltmate, B. "Combined Spatial Filtering and Boolean Operators Applied to the Processing of Real Images," MS Thesis, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, June 1982. (AFIT/GCS/EE/82J-8)

7. Rosenfeld, A. and A.C. Kak. Digital Picture Processing Vol 2, New York: Academic Press, 1982.

8. Hall, E.L. Computer Image Processing and Recognition. New York: Academic Press, 1979.

APPENDIX A

SUPPORT SOFTWARE

```
      SUBROUTINE IOF(N,MAIN,F1,F2,F3,MS,S1,S2,S3)
C
C     Written by Lt. Simmons          10 Sep 1981
C     Version 2
C
C     This FORTRAN 5 subroutine will read from the file
C     COM.CM (FCOM.CM in the foreground) the program name,
C     any global switches, and up to three local file
C     names and corresponding local switches.
C
C     Calling arguments:
C
C     N is the number of local files and switches to be
C     read from (F)COM.CM.  N must be 1, 2, or 3.
C
C     MAIN is an ASCII array for the main program file name.
C
C     F1, F2, and F3 are the three ASCII arrays to return
C     the local file names.
C
C     MS is a two-word integer array that holds any global
C     switches.
C
C     S1, S2, and S3 are two-word integer arrays that
C     hold the local switches corresponding to F1 through
C     F3 respectively.
C
C
C     Dimension the arrays.
C
      DIMENSION MAIN(7),MS(2)
      INTEGER F1(7),F2(7),F3(7),S1(2),S2(2),S3(2)
C
C     Check the bounds on N.
C
      IF(N.LT.1.OR.N.GT.3)STOP "N out of bounds in IOF."
C
C     Process the data in (F)COM.CM
C
      CALL GROUND(I)  ;Find out which ground program is in
      IF(I.EQ.0)OPEN 0,"COM.CM"        ;Open ch. 0 to COM.CM
      IF(I.EQ.1)OPEN 0,"FCOM.CM"       ;Open ch. 0 to FCOM.CM
      CALL COMARG(0,MAIN,MS,IER)       ;Read from (F)COM.CM
      IF(IER.NE.1)TYPE" COMARG error:",IER
      WRITE(10,1)MAIN(1)               ;Type program name
 1    FORMAT(' Program ',S13,'running.')
      CALL COMARG(0,F1,S1,JER)         ;Read from (F)COM.CM
      IF(JER.NE.1)TYPE" COMARG error (F1):",JER
      IF(N.EQ.1)GO TO 2                ;Test N
      CALL COMARG(0,F2,S2,KER)         ;Read from (F)COM.CM
      IF(KER.NE.1)TYPE" COMARG error (F2):",KER
      IF(N.EQ.2)GO TO 2                ;Test N
      CALL COMARG(0,F3,S3,LER)         ;Read from (F)COM.CM
      IF(LER.NE.1)TYPE" COMARG error (F3):",LER
 2    CLOSE 0
      RETURN
      END
```

```
      SUBROUTINE UNPACK(N,PIXWORD,PIXELS)
C
C     Written by Lt. Simmons          Version 2
C
C     This subroutine will unpack four 4-bit integers from a
C     16-bit integer word.  The pixels in a video file have to
C     be unpacked if each pixel is to be operated on separately.
C
      INTEGER PIXWORD(N),PIXELS(4,N)         ;Four pixels per word
      DO 1 I=1,N                            ;'N' allows higher-order
      DO 1 J=1,4                            ;arrays to be passed.
      PIXELS((5-J),I)=15.AND.PIXWORD(I)      ;Pick off right pixel
    1 PIXWORD(I)=ISHFT(PIXWORD(I),-4)       ;Shift word 4 bits right
      RETURN                                ;to pick off next pixel.
      END
```

```
      SUBROUTINE REPACK(N,PIXELS,PXWD)
C
C     Written by Lt. Simmons          Version 2
C
C     This subroutine will repack four 4-bit integer pixels
C     into one 16-bit word for use by CHOPS.  Parameter N
C     allows more than one 4-bit to 1-word repacking
C     operation in each call to REPACK.
C
      INTEGER PIXELS(4,N),PXWD(N)
      DO 1 J=1,N                    ;Loop N times
      PXWD(J)=0
      DO 1 I=1,4
      PXWD(J)=ISHFT(PXWD(J),4)      ;Shift pixel left in word
    1 PXWD(J)=PIXELS(I,J)+PXWD(J) ;then add next pixel on right
      RETURN
      END
```

```
C     Program PICT       Written by Lt. Simmons      14 Oct 1981
C     Fortran 5          Revised by Lt. Cromer       12 July 1982
C                        Revised by Capt Wells       25 Sept 1983

C     This program will convert video pixels to lineprinter pixels
C     and will send the picture to the Eclipse's Printronix 300
C     lineprinter.  This program prints the complete 256 by 256
C     pixel picture. Odd numbered video lines are represented by 3x3
C     pixels, while even numbered video lines are represented by 4x3
C     pixels. Run time should be about two minutes.

      DIMENSION I1ARRAY(268),I2ARRAY(264),I3ARRAY(201),I4ARRAY(198)
      EQUIVALENCE (I1ARRAY,I2ARRAY,I3ARRAY,I4ARRAY)
      DIMENSION ILP(4,67),IFILE(7),IREC(64),ISAV(4)
      INTEGER MAIN(7),F2(7),F3(7),MS(2),S1(2),S2(2),S3(2)

      CALL IOF(1,MAIN,IFILE,F2,F3,MS,S1,S2,S3)

C     Set up solid line, space, and line feed/plot-on character

      IL=177777K                                 ;Solid line
      NC=40100K                                  ;Space
      LF=012K                                    ;Line feed
      LFPC=2412K                                 ;Line feed/plot on

C     Set up parameters for complete picture display.

      N1=66                            ;Top and bottom border length
      N2=256                           ;Number of lines displayed
      N3=1                             ;Location of left border
      N4=66                            ;Location of right border
      N5=67                            ;Location of line feed
      N6=1                             ;Length of lines displayed

      CALL OPEN(1,IFILE,1,IER)                   ;Open the video file
      CALL CHECK(IER)

C     Put a border at the top of the picture.

    5 DO 7 I=1,3
        DO 6 J=1,N1
          WRITE BINARY(12)IL                      ;Print a line
    6   CONTINUE
        WRITE BINARY(12)LFPC                     ;Terminate the line
    7 CONTINUE

C     Each line of the picture will have a border on each end.

      JTEST = -1

      DO 13 JA=1,N2

        JTEST = JTEST *(-1)
        JJ = 4
        IF(JTEST.GT.0)JJ=3                             ;Odd iteration

C     Put a border down the left hand side

        DO 8 K=1,JJ
          ILP(K,N3)=43500K
```

```
    8     CONTINUE

C      Put a border down the right hand side

          DO 9 L=1,JJ                                      ;Insert border
             ILP(L,N4)=40170K                              ;line feed after
             ILP(L,N5)=LFPC                                ;the picture
    9     CONTINUE

C      Convert the video picture pixels to $LPT pixels

          READ BINARY(1)IREC           ;Read one video line

          DO 12 N=N6,64

             IWR=BYTE(IREC(N),2)      ;Right two pixels
             IWL=BYTE(IREC(N),1)      ;Left two pixels

             IF(JJ.GT.3.5)CALL OUT4X3(IWL,ISAV)
             IF(JJ.LT.3.5)CALL OUT3X3(IWL,ISAV)

             DO 11 JB=1,JJ
                BYTE(ILP(JB,N+1),1)=ISAV(JB)     ;Move to high byte
   11        CONTINUE

             IF(JJ.LT.3.5)CALL OUT3X3(IWR,ISAV)
             IF(JJ.GT.3.5)CALL OUT4X3(IWR,ISAV)

             DO 12 JC=1,JJ
                BYTE(ILP(JC,N+1),2)=ISAV(JC)     ;Store low byte
   12     CONTINUE

          L=0

          DO 130 JE=1,JJ
             DO 130 JD=1,N5
                L=L+1
                I1ARRAY(L)=ILP(JE,JD)
  130     CONTINUE

          IF(N5.EQ.66..AND.JJ.EQ.3)WRITE BINARY(12)I4ARRAY
          IF(N5.EQ.67..AND.JJ.EQ.3)WRITE BINARY(12)I3ARRAY
          IF(N5.EQ.66..AND.JJ.EQ.4)WRITE BINARY(12)I2ARRAY
          IF(N5.EQ.67..AND.JJ.EQ.4)WRITE BINARY(12)I1ARRAY

   13 CONTINUE

C      Put a border and title at the bottom of the picture

      DO 15 JF=1,3
         DO 14 JG=1,N1
            WRITE BINARY(12)IL                             ;Print a line
   14    CONTINUE
         WRITE BINARY(12)LFPC                           ;Terminate the line
   15 CONTINUE

      WRITE(12,16)                                  ;Title picture
   16 FORMAT(' ',15X,'Signal Processing Laboratory, Air Force
     +  Institute of Technology, Wright-Patterson AFB, OH  45433<14>')

      CALL RESET                                    ;Close all channels
      STOP
```

```
      SUBROUTINE OUT3X3(VIDPIX,LINEPRINT)
C*********************************************************************
C
C       Written by Lt. J.H. Cromer
C
C       This subroutine converts the video pixel values
C       (i.e. an integer value from 0-15) to lineprinter
C       dot matrix form.  A 3x3 array pattern
C       is formed by this subroutine.  Dot pattern texture
C       (distribution of dots) and average brightness are
C       varied to create 16 pseudo-gray levels.  Odd numbered
C       rows of the picture created by PICTURE use
C       these 3x3 patterns.
C       NOTE:  The six least significant bits of each byte
C                sent to the P-300 represent print hammer
C                switches (i.e. a 1 turns the hammer on
C                to print a dot, a 0 leaves it off).  Bit
C                seven must have a value of 1.
C       (See the Printronix manual for further discussion)
C
C********************************************************************
      INTEGER VIDPIX,LINEPRINT(3),RIGHT,PATTERN(3,16,2)
C
C       Note that right and left pixel patterns are
C       not necessarily the same.
C
      DATA PATTERN/4*7,5,7,6,7,3,7,5,2*2,7,2,5,2,5,2,7,2*2,
     $5,2,5,0,5,2,5,2,5,0,2*2,0,2*2,0,2,1,3*0,2,4*0,
     $4*170K,150K,170K,160K,170K,130K,120K,150K,170K,4*150K,
     $120K,150K,120K,170K,2*120K,150K,120K,150K,100K,150K,
     $140K,120K,110K,120K,100K,150K,110K,140K,110K,100K,150K,
     $2*100K,120K,7*100K/
      RIGHT=(VIDPIX.AND.15)+1
      LEFT=(ISHFT(VIDPIX,-4).AND.15)+1
      DO 10 I=1,3
            LINEPRINT(I)=PATTERN(I,LEFT,1)+PATTERN(I,RIGHT,2)
 10   CONTINUE
      RETURN
      END
C
C********* Subroutine OUT3X3 ***********************************
```

```
      SUBROUTINE OUT4X3(VIDPIX,LINEPRINT)
C*********************************************************************
C
C     Written by Lt. Cromer
C
C     This subroutine returns lineprinter pixels to the
C     calling program PICTURE, which sends video
C     pixels (an integer from 0-15).  The pixel pattern
C     returned is a 4x3 dot matrix array, to be used for
C     the even rows of pictures created by PICTURE.
C     (See OUT3X3.FR for more explanation).
C
C*********************************************************************
      INTEGER VIDPIX,LINEPRINT(4),RIGHT,PATTERN(4,16,2)
      DATA PATTERN/5*7,5,7,7,6,7,7,3,3,6,7,5,5,6,3,2,5,2,
     $3*5,2,2,5,5,0,2,5,2,2,5,2,2,1,4,2,2,4,1,2,1,4,2,0,0,1,
     $4,0,0,2,10*0,5*170K,150K,2*170K,160K,2*170K,2*130K,
     $160K,170K,150K,120K,2*170K,120K,150K,120K,3*150K,2*120K,
     $150K,120K,2*150K,3*120K,150K,2*120K,110K,140K,120K,150K,
     $2*100K,150K,120K,100K,110K,140K,120K,2*100K,
     $120K,2*100K,140K,3*100K,120K,5*100K/
      RIGHT=(VIDPIX.AND.15)+1
      LEFT=(ISHFT(VIDPIX,-4).AND.15)+1
      DO 10 I=1,4
            LINEPRINT(I)=PATTERN(I,LEFT,1)+PATTERN(I,RIGHT,2)
 10   CONTINUE
      RETURN
      END
C
C********* Subroutine OUT4X3 ************************************
```

```
C*************************************************************
C
C       Program TONER           Written by Lt. Jim Cromer
C
C       This program will convert individual pixel values
C       of an input video file into new values assigned by
C       the user.  It can be used to adjust gray-levels
C       (i.e. histogram equalization), increase contrast,
C       display selected pixel values, create "negative im-
C       ages", create files of a constant gray-level, and for
C       many other purposes. (NOTE: The program EVIDHIST can
C       be run to generate a histogram of any VIDEO file.)
C
C       Execution Line Format:
C               TONER
C               The program will ask for the input and output
C         file names, and the new pixel values.
C
C       Load Line Format:
C               RLDR TONER TIMER UNPACK REPACK @FLIB@
C
C*************************************************************
        INTEGER NEWVALUE(0:15),INFILE(7),OUTFILE(7)
        INTEGER PACKED(4096),UNPACKED(16384)
        ISTART=0                ;start timer
        ISTOP=1                 ;stop timer

C
C******** USER INPUT OF VARIABLES ****************************
C
        ACCEPT"Enter name of video file to be toned ---> "
        READ(11,1000)INFILE(1)
        ACCEPT"Enter name of output file ---> "
        READ(11,1000)OUTFILE(1)
        TYPE"<15>","Enter new pixel values<15>"
        TYPE"   NOTE:  Leading zeros are significant, i.e. enter"
        TYPE"   a '1' as '01', enter a '2' as '02', etc.<15>"
        DO 1 J=0,15             ;16 gray-levels
                WRITE(10,2000)J
                READ(11,3000)NEWVALUE(J)
                IF(NEWVALUE(J).LT.0.OR.NEWVALUE(J).GT.15)
     $NEWVALUE(J)=15
 1      CONTINUE
        CALL TIMER(ISTART)
C
C******** I/O FILE MANAGEMENT ********************************
C
        CALL OPEN(1,INFILE,1,IER)
        IF(IER.NE.1)TYPE"INFILE OPEN error #",IER
        CALL DFILW(OUTFILE,IER)
        IF(IER.NE.1.AND.IER.NE.13)TYPE"OUTFILE DFILW
     $ error #",IER
        CALL CFILW(OUTFILE,3,64,IER)    ;create a contiguous file
        IF(IER.EQ.41)CALL CFILW(OUTFILE,2,IER)
```

```
      IF(IER.NE.1)TYPE"OUTFILE CFILW error #",IER
      CALL OPEN(2,OUTFILE,3,IER)
      IF(IER.NE.1)TYPE"OUTFILE OPEN error #",IER
C
C******** RE-TONE THE PICTURE *********************************
C
      DO 3 J=0,48,16
              CALL RDBLK(1,J,PACKED,16,IER)
              IF(IER.NE.1)TYPE"RDBLK #",J," error:",IER
              CALL UNPACK(4096,PACKED,UNPACKED)
              DO 2 I=1,16384   ;do 1/4 of picture
                      UNPACKED(I)=NEWVALUE(UNPACKED(I))
 2            CONTINUE
              CALL REPACK(4096,UNPACKED,PACKED)
              CALL WRBLK(2,J,PACKED,16,IER)
              IF(IER.NE.1)TYPE"WRBLK #",J," error:",IER
 3    CONTINUE
C
C     Send message to CRT terminal
C
      CALL TIMER(ISTOP)
      WRITE(10,4000)OUTFILE(1)
      TYPE"<15>","Have a nice day!<7><15>"
C
C********* WRITE NEW TONE VALUES TO THE LINEPRINTER ***************
C
      WRITE(12,5000)
      WRITE(12,6000)INFILE(1),OUTFILE(1)
      DO 5 I=0,15
              WRITE(12,7000)I,NEWVALUE(I)
 5    CONTINUE
 1000 FORMAT(S13)
 2000 FORMAT(" Change old pixel value",I3," to ?")
 3000 FORMAT(I2)
 4000 FORMAT(" The toned picture is in the file --->  ",S13)
 5000 FORMAT(//////26X," RESULTS OF TONER<10>"/
     $26X," ----------------"////)
 6000 FORMAT(10X," Input file ---> ",S13,/10X," Output file
     $---> ",S13,//20X,"OLD PIXEL",10X,"NEW PIXEL",/20X,
     $"---------",10X,"---------"/)
 7000 FORMAT(23X,I2,17X,I2)
      CALL RESET
      STOP
      END
C
C*********** Program TONER *******************************************
```

```
      SUBROUTINE TIMER(I)
C*****************************************************************
C
C       Subroutine TIMER        Written by  Lt. Jim Cromer
C       Fortran 5
C
C       This subroutine is used to time the real-time execution
C       time of the calling program.  If the parameter passed, I,
C       is equal to 0, the timer is unconditionally started.
C       If I is not equal to 0, the timer is unconditionally
C       stopped, and the total run time is typed on the console
C       CRT.
C
C       Execution Line Format
C               CALL TIMER(I)   ;IF(I.EQ.0), start timing
C                               ;IF(I.NE.0), stop timing
C
C*****************************************************************
      COMMON /ITIME/ IH1,IM1,IS1
      IF(I.NE.0)GO TO 100
      CALL FGTIME(IH1,IM1,IS1)          ;get starting time
      WRITE(10,1000)IH1,IM1,IS1
 1000 FORMAT(//" START TIME --->",I4,":",I3,":",I3)
      RETURN
  100 CALL FGTIME(IH2,IM2,IS2)          ;get stopping time
      WRITE(10,2000)IH2,IM2,IS2
 2000 FORMAT(//"  STOP TIME --->",I4,":",I3,":",I3)
      ITOTAL=3600*(IH2-IH1)+60*(IM2-IM1)+IS2-IS1
      HOURS=INT(ITOTAL/3600)
      TRON=(ITOTAL-3600*HOURS)          ;intermediate variable
      MINS=INT(TRON/60)
      ISECS=MOD(TRON,60)
      WRITE(10,3000)HOURS,MINS,ISECS
 3000 FORMAT(//" TOTAL TIME --->",I4,":",I3,":",I3)
      RETURN
      END
C
C************** Subroutine TIMER ********************************
```

```
C*********************************************************************
C
C       Program NMOVE            Written by Lt. Jim Cromer
C       Fortran IV               16 Aug 1982
C
C       This program will place the video information in the win-
C       dow given for the template (inputfile1) inside of the
C       window given for the background (inputfile2), and write
C       the combined picture to the outputfile.  The window may be
C       placed anywhere within the background, and may be taken
C       from anywhere within the template.  Window width, length,
C       and position are input by the user.
C
C       Execution line format:  (on the NOVA only)
C               NMOVE            (run EMOVE on the ECLIPSE)
C
C       Loader command line format (NOVA only):
C       RLDR NMOVE TEST BLOCK CHANGE XWRBLK XRDBLK
C               UNPACK REPACK FORT.LB
C
C*********************************************************************
C
        INTEGER IPAR(2),INFILE1(7),INFILE2(7),OUTFILE(7),
     $CB1,CBLOCKS,CCOL,CLS,CSTOP,CTOP,CLB,CLEFT,TTOP,TB1,TBLOCKS,CH3,
     $TCOL,TLS,TSTOP,TLB,TLEFT,COMB(1024),TEMP(1024),BACK(1024),WIDTH,TB
        COMMON/LIST1/COMB,TEMP,CLS,TLS
        COMMON/LIST2/LENGTH,WIDTH
        EQUIVALENCE (COMB,BACK)
C
C
C******* I/O FILE MANAGEMENT ************************************
C
        TYPE"<15>","Program NMOVE is to be run on the NOVA only!"
 99     TYPE"<15>","*****************************************<15>"
        ACCEPT" Enter template file name: "
        READ(11,1000)INFILE1(1)
        ACCEPT"<15>"," Enter background file name: "
        READ(11,1000)INFILE2(1)
        DO 999 J=1,7
 999            OUTFILE(J)=INFILE2(J)
        ACCEPT"<15>"," Enter combined output file name: "
        READ(11,1000)OUTFILE(1)
 1000   FORMAT(S13)
        CALL OPEN(1,INFILE1,2,IER)
        IF(IER.NE.1)TYPE" Channel 1 OPEN error:",IER
        CALL OPEN(2,INFILE2,2,IER)
        IF(IER.NE.1)TYPE" Channel 2 OPEN error:",IER
        CH3=2
        ICOUNT=0
        DO 1002 J=1,7              ;check for BACKGROUND=COMBINED
 1002           IF(OUTFILE(J).EQ.INFILE2(J))ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.7)GO TO 1
        CH3=3
        CALL DFILW(OUTFILE,IER)
        IF(IER.NE.1.AND.IER.NE.13)TYPE"OUTFILE DFILW error:",IER
        CALL CFILW(OUTFILE,2,IER)
        IF(IER.NE.1)TYPE" CFILW error:",IER
        CALL OPEN(CH3,OUTFILE,2,IER)
        IF(IER.NE.1)TYPE" Channel 3 OPEN error:",IER
```

```
C******* ENTER WINDOW PARAMETERS ********************************
C
  1     ACCEPT"<15>"," Enter top row of template window (1-256):",TTOP
        ACCEPT" Enter left column of template window (1-256):",TLEFT
        ACCEPT" Enter width of window (1-256):",WIDTH
        ACCEPT" Enter length of window (1-256):",LENGTH
C
C       The calls to TEST check to see if the input parameters are
C       legal, and modifies them if necessary:
C               0 < TOP < 257,     (TOP + LENGTH) < 258
C               0 < LEFT < 257,    (LEFT + WIDTH) < 258
C
        CALL TEST(TTOP,TLEFT)
        ACCEPT"<15>"," Enter top row of background window (1-256):",CTOP
        ACCEPT" Enter left column of background window (1-256):",CLEFT
        CALL TEST(CTOP,CLEFT)
        CALL BLOCK(TBLOCKS,TB1,TLS,TCOL,TTOP,TLEFT)
        CALL BLOCK(CBLOCKS,CB1,CLS,CCOL,CTOP,CLEFT)
C
C       Determine column number of the last video row (0-3)
C
        J1=MOD(LENGTH,4)
        TSTOP=MOD((TCOL+J1),4)-1
        CSTOP=MOD((CCOL+J1),4)-1
        IF(CSTOP.EQ.-1)CSTOP=3
        IF(TSTOP.EQ.-1)TSTOP=3
C
C       Determine the last significant block of window
C
        CLB=CB1+CBLOCKS-1
        TLB=TB1+TBLOCKS-1
C
C       User check of window parameters
C
        TYPE"<15>","WIDTH=",WIDTH,"      LENGTH=",LENGTH
        TYPE"TEMPLATE TOP ROW=",TTOP,"  BACKGROUND TOP ROW=",CTOP
        TYPE"TEMPLATE LEFT COLUMN=",TLS,"  BACKGROUND LEFT COLUMN
     $=",CLS,"<15>"
        ACCEPT"Enter 1 to see expanded set of variables, any
     $ other integer to continue:  ",I
        IF(I.NE.1)GO TO 5

        TYPE"************************************************"
        TYPE" PARAMETER    TEMPLATE  BACKGROUND"
        TYPE" __________   ________  __________"
        TYPE"<15>"," TOP ROW    ",TTOP,CTOP
        TYPE"<15>"," START COL #",TCOL,CCOL
        TYPE"<15>"," STOP COL # ",TSTOP,CSTOP
        TYPE"<15>"," FIRST BLOCK",TB1,CB1
        TYPE"<15>"," LAST BLOCK ",TLB,CLB
        TYPE"<15>"," # OF BLOCKS",TBLOCKS,CBLOCKS
        TYPE"<15>"," LEFT COL   ",TLS,CLS
        TYPE"<15>"," WIDTH= ",WIDTH
        TYPE"<15>"," LENGTH=",LENGTH
        TYPE"<15>","************************************************"
        ACCEPT" Enter 1 to try another set, any other integer
     $ to continue: ",I
        IF(I.EQ.1)GO TO 1
```

```
C****** Create the combined picture **************************
C
  5     ICOUNT=0
        IF(CH3.EQ.2)GO TO 20    ;If combined picture file
                                ;is the same as the back-
                                ;ground picture file, then no
                                ;need to write to itself
C       Write background only blocks (before window)
C       to the combined picture file.
C
        JMAX=CB1-1
        IF(JMAX.LT.0)GO TO 20
        DO 10 J=0,JMAX
                CALL RDBLK(2,J,BACK,1,IER)
                IF(IER.NE.1)TYPE" 2RDBLK",J," error:",IER
                CALL WRBLK(CH3,J,COMB,1,IER)
                IF(IER.NE.1)TYPE" WRBLK",J," error:",IER
                ICOUNT=ICOUNT+1
 10     CONTINUE
 20     TYPE" Background before window completed."
        TYPE" # Blocks written:",ICOUNT
C
C
C . . .  Overlay template window onto background  . . . . .
C
        CALL XRDBLK(1,TB1,TEMP,1,IER)
        IF(IER.NE.1)TYPE"1RDBLK #",TB1," error:",IER
        CALL XRDBLK(2,CB1,BACK,1,IER)
        IF(IER.NE.1)TYPE"2RDBLK #",CB1," error:",IER
        N1=TCOL           ;4-MAX(N1,N2) gives the number of rows
        N2=CCOL           ;to change before the next RDBLK
        IF(TCOL.GT.CCOL)GO TO 100
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C       There are four columns in the packed video array (64x4),
C       designated 0, 1, 2, and 3.  If the template starting
C       column number is less than or equal to the background (combined)
C       starting column number, then the background block will be "used
C       up" before the template block.  When the background block is
C       finished, a WRBLK is done, and the next background block is read.
C       When the template block is finished, the next template block is
C       read, but no WRBLK needs to be performed.  Note that the back-
C       ground and combined files are always at the same block number.
C
        CALL CHANGE(N2,N2,N1)
        CALL XWRBLK(CH3,CB1,COMB,1,IER)
        IF(IER.NE.1)TYPE" WRBLK #",CB1," error:",IER
C
C       Write the template window into the background
C
        TB=TB1+1
        IMIN=CB1+1
        ICOUNT=1
        IMAX=CLB-1
        IF(IMIN.GT.IMAX)GO TO 60
        DO 50 I=IMIN,IMAX
                CALL XRDBLK(2,I,BACK,1,IER)
                IF(IER.NE.1)TYPE" 2RDBLK #",I," error:",IER
                CALL CHANGE(N1,N2,N1)
                CALL XRDBLK(1,TB,TEMP,1,IER)
                IF(IER.NE.1)TYPE" 1RDBLK #",TB," error:",IER
                CALL CHANGE(N2,N2,N1)
```

```
                CALL XWRBLK(CH3, I, COMB, 1, IER)
                IF(IER.NE.1)TYPE" WRBLK #", I, " error: ", IER
                ICOUNT=ICOUNT+1
 50     TB=TB+1
 60     TYPE" TCOL.LT.CCOL--Window portion complete."
        TYPE" # blocks written: ", ICOUNT
        GO TO 250
C
C       In this case the template starting column number
C       is greater than the background starting column number.
C       The template block must be "finished" first.
C
100     CALL CHANGE(N1,N2,N1)            ;finish TEMP block
        TB=TB1+1
        IMAX=CLB-1
        ICOUNT=0
        IF(CB1.GT.IMAX)GO TO 225
        DO 200 I=CB1,IMAX
                CALL XRDBLK(1,TB,TEMP,1,IER)
                IF(IER.NE.1)TYPE" 1RDBLK #",TB," error: ",IER
                CALL CHANGE(N2,N2,N1)   ;finish BACK block
                CALL XWRBLK(CH3,I,COMB,1,IER)
                IF(IER.NE.1)TYPE" WRBLK #",I," error: ",IER
                ICOUNT=ICOUNT+1
                IBLK=I+1
                CALL XRDBLK(2,IBLK,BACK,1,IER)
                IF(IER.NE.1)TYPE" 2RDBLK #",IBLK," error: ",IER
                CALL CHANGE(N1,N2,N1)   ;finish TEMP block
200     TB=TB+1
225     TYPE" TCOL.GT.CCOL--Window portion complete."
        TYPE" # blocks written: ",ICOUNT
C
C
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C       If the combined (background) stopping column number
C       is greater than the template stop column number, then the
C       second last template block (I2LB=TLB-1) must be read (i.e.
C       there are more video rows to be changed in the last back-
C       ground block than there are available in the last template
C       block to change them to).  If TSTOP is greater than or equal
C       to CSTOP, then there are sufficient video rows available in
C       the last template block to complete the last
C       background block to be changed.
C
250     IF(CSTOP.GT.TSTOP)GO TO 400
        M1=TSTOP-CSTOP
        CALL XRDBLK(1,TLB,TEMP,1,IER)
        IF(IER.NE.1)TYPE" 1RDBLK #",TLB," error: ",IER
        CALL XRDBLK(2,CLB,BACK,1,IER)
        IF(IER.NE.1)TYPE" 2RDBLK #",CLB," error: ",IER
        N1=3-CSTOP
        N2=0
        CALL CHANGE(N1,N2,M1)   ;finish BACK block to CSTOP
        CALL XWRBLK(CH3,CLB,COMB,1,IER)
        IF(IER.NE.1)TYPE" WRBLK #",CLB," error: ",IER
        TYPE" CSTOP.LT.TSTOP--Last block of window complete."
        GO TO 500
C
C       Complete the last block of the window
C       NOTE:  CSTOP is greater than TSTOP.  Therefore finish
C              TEMP before BACK.
C
400     M1=CSTOP-TSTOP
```

```
        CALL XRDBLK(2,CLB,BACK,1,IER)
        IF(IER.NE.1)TYPE" 2RDBLK #",CLB," error:",IER
        I2LB=TLB-1
        CALL XRDBLK(1,I2LB,TEMP,1,IER)
        IF(IER.NE.1)TYPE" 1RDBLK #",I2LB," error:",IER
        N1=4-M1
        N2=0
        CALL CHANGE(N1,N2,N1)             ;finish TEMP block
        CALL XRDBLK(1,TLB,TEMP,1,IER)
        IF(IER.NE.1)TYPE" 1RDBLK #",TLB," error:",IER
        M1=3-TSTOP
        CALL CHANGE(M1,N2,N1)             ;finish BACK block to CSTOP
        CALL XWRBLK(CH3,CLB,COMB,1,IER)
        IF(IER.NE.1)TYPE" WRBLK #",CLB," error:",IER
        TYPE" CSTOP.GT.TSTOP--Last block of window complete."
  500   ICOUNT=1
C
C       Finish the combined file (background only portion)
C
        JMIN=CLB+1
        IF(JMIN.GT.63)GO TO 601           ;if finished STOP
        IF(CH3.EQ.2)GO TO 601             ;if COMBINED=BACKGROUND, STOP
        DO 600 J=JMIN,63
                CALL RDBLK(2,J,BACK,1,IER)
                IF(IER.NE.1)TYPE" 2RDBLK #",J," error:",IER
                CALL WRBLK(CH3,J,COMB,1,IER)
                IF(IER.NE.1)TYPE" WRBLK #",J," error:",IER
                ICOUNT=ICOUNT+1
  600   CONTINUE
        TYPE" Finished background only portion."
  601   TYPE" # blocks written:",ICOUNT
        TYPE"<15>","<7>","<15>"," Program NMOVE execution completed. <7>"
        WRITE(10,2000)OUTFILE(1)
 2000   FORMAT(" The combined picture is in the file --> ",S13)
C
C
C****** Present Option Menu **********************************
C
        GO TO 2010
 2002   TYPE"<15>","Input error.  Try again."
 2010   TYPE"<15>","*******************************************"
        TYPE"<15>","What next?<15>","Here are the options:"
        TYPE"<15>","<11>1 - Try another set of window values"
        TYPE"<15>","<11>2 - Start over with new input pictures"
C       TYPE"<15>","<11>3 - Display combined picture on the video monitor"
        TYPE"<15>","<11>3 - Save combined picture and STOP<15>"
        ACCEPT"<11>Enter option ---> ",IOPT
        IF(IOPT.EQ.1)GO TO 1
        CALL RESET
        IF(IOPT.EQ.2)GO TO 99
        IF(IOPT.EQ.3)STOP
        TYPE"<15>","Check monitor - - Press green CHOPS control
     $button to continue."
        IDCNT=4
        IPAR(1)=9999
        IPAR(2)=0
        WRITE(10,3000)OUTFILE(1)
 3000   FORMAT("0","Picture being displayed  ---> ",S13)
C       CALL CHANNEL(0,0,3,0,0,"A",0,0,0,IE,IS)          ;call abort
C       CALL CHANNEL(3,1,2,1,IDCNT,OUTFILE,64,0,IPAR,IERR,ISYS)
C       CALL ERCHK(IERR,1,IDCNT,1,ISYS)
        TYPE"<15>","CHANNEL currently not loaded."
        TYPE"Use VIDEO to display combined pictures.<15>"
        CALL OPEN(1,INFILE1,2,IER)                ;re-OPEN channels
        IF(IER.NE.1)TYPE"CH1 RE-OPEN ERROR:",IER
        CALL OPEN(2,INFILE2,2,IER)
        IF(IER.NE.1)TYPE"CH2 RE-OPEN ERROR:",IER
        IF(CH3.EQ.3)CALL OPEN(3,OUTFILE,2,IER)
        IF(IER.NE.1)TYPE"CH3 RE-OPEN ERROR:",IER
        GO TO 2010
        END
```

```
      SUBROUTINE TEST(TOP,LEFT)
C*******************************************************************
C
C     Subroutine TEST checks to see if the input parameters
C     to program NMOVE are legal, and modifies them if
C     necessary.  (It is also called by DISTANCE.)
C
C*******************************************************************
      INTEGER TOP,WIDTH
      COMMON/LIST2/LENGTH,WIDTH
      IF(LEFT.LT.1.OR.LEFT.GT.256)LEFT=1
      MAXWIDTH=257-LEFT               ;picture has 256 columns
      IF(WIDTH.GT.MAXWIDTH.OR.WIDTH.LT.1)WIDTH=MAXWIDTH
      IF(TOP.LT.1.OR.TOP.GT.256)TOP=1
      MAXLENGTH=257-TOP               ;picture has 256 rows
      IF(LENGTH.GT.MAXLENGTH.OR.LENGTH.LT.1)LENGTH=MAXLENGTH
      RETURN
      END
C
C********* Subroutine TEST *****************************************
```

```
      SUBROUTINE BLOCK(NUMBLOCKS,BLOCK1,LEFTSIDE,COLUMN,TOP,LEFT)
C***********************************************************************
C
C     Subroutine BLOCK determines the total number of blocks to
C     be read into the window, the first block to be read,
C     and the first video row "column" number.  This subroutine
C     is called by NMOVE and DISTANCE.
C
C***********************************************************************
      INTEGER BLOCK1,COLUMN,TOP,REMAINDER,WIDTH
      COMMON/LIST2/LENGTH,WIDTH
      BLOCK1=INT((TOP-1)/4.0)         ;4 rows per block
      COLUMN=MOD((TOP-1),4)
      LEFTSIDE=LEFT
      REMAINDER=MOD(LENGTH,4)
      K1=LENGTH+3
      NUMBLOCKS=INT(K1/4.0)
      IF(REMAINDER.EQ.2.AND.COLUMN.GT.2)NUMBLOCKS=NUMBLOCKS+1
      IF(REMAINDER.EQ.3.AND.COLUMN.GT.1)NUMBLOCKS=NUMBLOCKS+1
      IF(REMAINDER.EQ.0.AND.COLUMN.GT.0)NUMBLOCKS=NUMBLOCKS+1
      IF(NUMBLOCKS.GT.1)RETURN
      TYPE"WARNING:  # Blocks to be read =",NUMBLOCKS
      PAUSE
      RETURN
      END
C
C*********** Subroutine BLOCK ***********************************
```

```
      SUBROUTINE CHANGE(JMIN,CSTART,TSTART)
C**********************************************************************
C
C     Written by Lt. Jim Cromer
C     Subroutine CHANGE changes the corresponding background
C     (i.e. the combined picture) pixels to template pixels;
C     it is called by the program NMOVE.
C
C**********************************************************************
      INTEGER COMB(1024),TEMP(1024),CLS,TLS,CSTART,TSTART,WIDTH
      COMMON /LIST2/ LENGTH,WIDTH
      COMMON /LIST1/ COMB,TEMP,CLS,TLS
C
      DO 2 J=JMIN,3
            K=TSTART*256+TLS          ;Set left side of input(template)
            M=CSTART*256+CLS          ;and output (combined) windows.
            KMAX=K+WIDTH-1  ;Change values over the width of window
            DO 1 L=K,KMAX
                  COMB(M)=TEMP(L)
 1                M=M+1
      TSTART=TSTART+1
 2    CSTART=CSTART+1
      IF(CSTART.EQ.4)CSTART=0          ;reset row-pointer if necessary
      IF(TSTART.EQ.4)TSTART=0
      RETURN
      END
C
C********* Subroutine CHANGE ******************************************
```

```
      SUBROUTINE XRDBLK(CH,J,FILE,I,IER)
C**********************************************************************
C
C     by Lt.  Jim Cromer
C     Subroutine XRDBLK performs a RDBLK to the designated
C     channel, reads a packed video file block, and
C     returns an unpacked array.
C
C**********************************************************************
      INTEGER CH,FILE(1024),VIDEO(256)
      K=256*I
      IF(J.GE.0.AND.J.LE.63)GO TO 1
      TYPE"ERROR: <7>BLOCK POINTER OUT OF BOUNDS IN XRDBLK"
      TYPE"          J=",J
      STOP
 1    IF(I.EQ.1)GO TO 2
      TYPE"ERROR IN <7>XRDBLK"
      TYPE" # Blocks to be read =",I
      STOP
 2    CALL RDBLK(CH,J,VIDEO,I,IER)
      DO 3 L=1,K
      DO 3 M=1,4
              ICOUNT=5-M+(L-1)*4
              FILE(ICOUNT)=15.AND.VIDEO(L)
              VIDEO(L)=ISHFT(VIDEO(L),-4)
 3    CONTINUE
      RETURN
      END
C
C********* Subroutine XRDBLK ************************************
```

```
      SUBROUTINE XWRBLK(CH,J,FILE,I,IER)
C
C     Given an unpacked video file, Subroutine XWRBLK
C     will pack the array and write it to the designated
C     unit number.
C
      INTEGER CH,FILE(1024),VIDEO(256)
      K=256*I
      CALL REPACK(K,FILE,VIDEO)
      CALL WRBLK(CH,J,VIDEO,I,IER)
      RETURN
      END
```

```
C     PROGRAM NCOMB - DG FORTRAN 5 - BY CAPT ROBERT WELLS, DEC 1983

C     THIS PROGRAM COMBINES UP TO FOUR VIDEO IMAGES (PACKED VIDEO
C     FORMAT). EACH PIXEL OF THE INPUT IMAGES IS TESTED TO SEE IF
C     IT IS NON-ZERO. THE VALUE OF THE FIRST NON-ZERO PIXEL OF THE
C     INPUT IMAGES IS ASSIGNED TO THE OUTPUT IMAGE AT THAT PIXEL
C     LOCATION. IF ALL INPUT IMAGES ARE ZERO, THE OUTPUT IS ZERO.
C     THE OUTPUT IS STORED IN PACKED VIDEO FORMAT.

      INTEGER INFNM1(7),INFNM2(7),INFNM3(7),INFNM4(7),OUTFNM(7)
      INTEGER IN1(2048),IN2(2048),IN3(2048),IN4(2048),OUT(2048)
      INTEGER TEMP(512),FLCNT

    1 FORMAT(S13)

    2 ACCEPT"ENTER NUMBER OF FILES TO BE COMBINED ->",FLCNT
      IF((FLCNT.LT.2).OR.(FLCNT.GT.4))GO TO 2

      GO TO (2,4,3) FLCNT

      ACCEPT"ENTER INPUT FILENAME ->"
      READ(11,1)INFNM4(1)
      CALL OPEN(5,INFNM4,1,IER)
      CALL CHECK(IER)

    3 ACCEPT"ENTER INPUT FILENAME ->"
      READ(11,1)INFNM3(1)
      CALL OPEN(4,INFNM3,1,IER)
      CALL CHECK(IER)

    4 ACCEPT"ENTER INPUT FILENAME ->"
      READ(11,1)INFNM2(1)
      CALL OPEN(3,INFNM2,1,IER)
      CALL CHECK(IER)

      ACCEPT"ENTER INPUT FILENAME ->"
      READ(11,1)INFNM1(1)
      CALL OPEN(2,INFNM1,1,IER)
      CALL CHECK(IER)

      ACCEPT"ENTER OUTPUT FILENAME ->"
      READ(11,1)OUTFNM(1)
      CALL OPEN(1,OUTFNM,3,IER)
      CALL CHECK(IER)

      DO 14 I=0,31   ; COUNT FOR 2K BUFFER LOADS

         GO TO (2,6,5) FLCNT

         CALL RDBLK(5,2*I,TEMP,2,IER)
         CALL CHECK(IER)
         CALL UNPACK(512,TEMP,IN4)

    5    CALL RDBLK(4,2*I,TEMP,2,IER)
         CALL CHECK(IER)
         CALL UNPACK(512,TEMP,IN3)

    6    CALL RDBLK(3,2*I,TEMP,2,IER)
         CALL CHECK(IER)
         CALL UNPACK(512,TEMP,IN2)
```

```
      CALL RDBLK(2,2*I,TEMP,2,IER)
      CALL CHECK(IER)
      CALL UNPACK(512,TEMP,IN1)

      DO 13 J=1,2048 ; COMBINING LOOP

         GO TO (2,8,7) FLCNT

         IF(IN4(J).NE.0)GO TO 9

 7       IF(IN3(J).NE.0)GO TO 10

 8       IF(IN2(J).NE.0)GO TO 11

         IF(IN1(J).NE.0)GO TO 12

         OUT(J)=0 ;ZERO IF ALL IN-FILES ARE ZERO
         GO TO 13

 9       OUT(J)=IN4(J)
         GO TO 13

10       OUT(J)=IN3(J)
         GO TO 13

11       OUT(J)=IN2(J)
         GO TO 13

12       OUT(J)=IN1(J)

13    CONTINUE

      CALL REPACK(512,OUT,TEMP)
      CALL WRBLK(1,2*I,TEMP,2,IER)
      CALL CHECK(IER)

14 CONTINUE

   CALL RESET
   STOP"<7><7><7><7>NCOMB"
   END
```

Notes on program INVERSE

Use: Two-dimensional inverse discrete fourier transform

Execution format: From CLI, command line is:

    INVERSE inputfile/I [outputfile/O] #/N

Parameter explanations:

Inputfile: The input file must be in binary format, since a RDBLK is used to read data from the file. The file is assumed to contain complex data, must be square, and must have a minimum length of 64. If the N by N array has N < 64, the columns should be augmented with zeros so that each column can be stored as one block of data. Thus, for N < 64, the size of the input array must be 64 rows by N columns of complex DFT samples. The origin of the frequency axes must be at (N/2 + 1, N/2 + 1).

Outputfile: The output file specification is optional. If no output file is specified, the inverse-transformed data is written back into the input file, overwriting it. The output data is complex and binary. Note: this program runs MUCH faster if Outputfile is NOT specified.

#: An integer must be specified to indicate the size of the input array (N by N), which is the size of the DFT computed. N must be a power of two, N < 512.

This program can be called from a DG fortran program since its last line is a call to FBACK. For more information, see the DG Fortran IV User's Manual, chapter II-5.

Notes on program DIRECT

Use: Two-dimensional discrete fourier transform

Execution format: From CLI, command line is:

```
DIRECT inputfile/I #/N [outputfile/O]
```

Parameter explanations:

Inputfile: The input file must be in binary format since a RDBLK is used to read data from the input file. The input file is assumed to contain complex data, with the imaginary part zero. The input array is assumed to be square, with a minimum row or column length of 64. If an array of size N < 64 is to be input, each column should be stored as one block of data (i. e. 64 complex points) even though only the first N complex numbers in the block correspond to actual data points. Only N rows need be stored. Thus, for N < 64, the size of the input array must be 64 by N columns.

#: An integer must be specified to indicate the size of the input array (N by N), which is the size of the DFT computed. N must be a power of two, N < 512.

Outputfile: The output file specification is optional. If no output file is specified, the transformed data will be written back into the input file, destroying the input data. The output file is complex, and is in binary format. The origin of the DFT samples will be at location (N/2, N/2), and the rows and columns will be transposed. Note: this program runs MUCH faster if Outputfile is NOT specified.

This program can be chained from a DG fortran program, since its last line is a call to FBACK. For more information, see the DG Fortran IV User's Manual, chapter II-5.

```
C     PROGRAM CMULT - DG FORTRAN 5 - CAPT ROBERT WELLS, DEC 1983

C     THIS PROGRAM PERFORMS COMPLEX CONJUGATE MULTIPLICATION OF TWO
C     256 X 256 COMPLEX FILES ON A POINT BY POINT BASIS.

      INTEGER I1FLNM(7),I2FLNM(7),OFLNM(7)
      INTEGER MAIN(7),MS(2),S1(2),S2(2),S3(2)
      COMPLEX IN1(2048),IN2(2048),OUT(2048)

      CALL IOF(3,MAIN,I1FLNM,I2FLNM,OFLNM,MS,S1,S2,S3)

      CALL OPEN(1,I1FLNM,1,IER)
      CALL CHECK(IER)
      CALL OPEN(2,I2FLNM,1,IER)
      CALL CHECK(IER)
      CALL OPEN(3,OFLNM,3,IER)
      CALL CHECK(IER)

      DO 3 I=0,31

         CALL RDBLK(1,32*I,IN1,32,IER)
         CALL CHECK(IER)
         CALL RDBLK(2,32*I,IN2,32,IER)
         CALL CHECK(IER)

         DO 2 J=1,2048
            OUT(J)=IN1(J)*CONJG(IN2(J))
    2    CONTINUE

         CALL WRBLK(3,32*I,OUT,32,IER)
         CALL CHECK(IER)

    3 CONTINUE

      CALL RESET
      STOP CMULT
      END
```

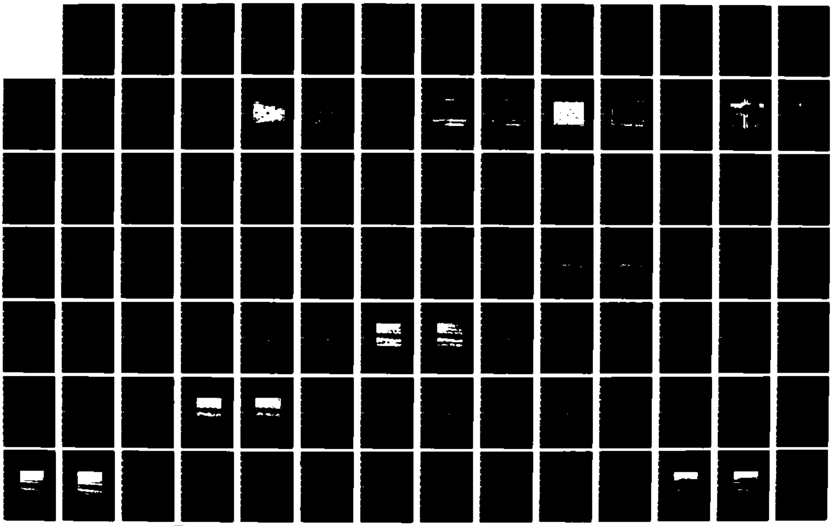
```
C       PROGRAM ADDCMP - DG FORTRAN - CAPT ROBERT WELLS, DEC 1983

C       THIS PROGRAM ADDS TWO COMPLEX FILES TO GET ONE COMPLEX FILE.
C       THE FILES MUST BE 256 X 256 COMPLEX FILES.

        INTEGER AFLNM(7),BFLNM(7),CFLNM(7),MAIN(7),MS(2),S1(2),S2(2),S3(2)
        COMPLEX AMAT(256,4),BMAT(256,4),CMAT(256,4)

        CALL IOF(3,MAIN,AFLNM,BFLNM,CFLNM,MS,S1,S2,S3)

        CALL OPEN(1,AFLNM,1,IER)
        CALL CHECK(IER)
        CALL OPEN(2,BFLNM,1,IER)
        CALL CHECK(IER)
        CALL OPEN(3,CFLNM,3,IER)
        CALL CHECK(IER)

        DO 2 I=0,63

           CALL RDBLK(1,16*I,AMAT,16,IER)
           CALL CHECK(IER)
           CALL RDBLK(2,16*I,BMAT,16,IER)
           CALL CHECK(IER)

           DO 1 K=1,4
              DO 1 J=1,256

              CMAT(J,K)=AMAT(J,K)+BMAT(J,K)

    1      CONTINUE

           CALL WRBLK(3,16*I,CMAT,16,IER)
           CALL CHECK(IER)

    2 CONTINUE

        CALL RESET
        STOP"<7><7><7><7>ADDCMP"
        END
```

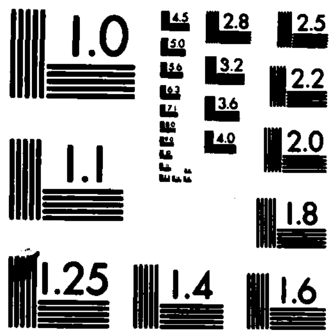
```
C     PROGRAM VDTOCP - DG FORTRAN 5 - CAPT ROBERT WELLS, DEC 1983

C     THIS PROGRAM CONVERTS A 256 X 256 PACKED VIDEO FILE TO A
C     256 X 256 COMPLEX FILE

      INTEGER IFLNM(7),OFLNM(7)
      INTEGER MAIN(7),F3(7),MS(2),S1(2),S2(2),S3(2)
      INTEGER TEMP(512),IN(2048)
      COMPLEX OUT(2048)

      CALL IOF(2,MAIN,IFLNM,OFLNM,F3,MS,S1,S2,S3)

      CALL OPEN(1,IFLNM,1,IER)
      CALL CHECK(IER)
      CALL OPEN(2,OFLNM,3,IER)
      CALL CHECK(IER)

      DO 3 I=0,31

         CALL RDBLK(1,2*I,TEMP,2,IER)
         CALL CHECK(IER)
         CALL UNPACK(512,TEMP,IN)

         DO 2 J=1,2048
            OUT(J)=CMPLX(FLOAT(IN(J)),0.0)
    2    CONTINUE

         CALL WRBLK(2,32*I,OUT,32,IER)
         CALL CHECK(IER)

    3 CONTINUE

      CALL RESET
      STOP VDTOCP
      END
```

```
C      PROGRAM CPTOVD - DG FORTRAN 5 - CAPT ROBERT WELLS, DEC 1983

C      THIS PROGRAM FINDS THE MAXIMUM AND MINIMUM OF A 256 X 256
C      COMPLEX FILE AND GIVES THE LOCATION OF THE MAXIMUM. ALSO, THE
C      PROGRAM CONVERTS THE FILE TO PACKED VIDEO FORMAT BY PERFORMING
C      A LINEAR SCALING TO THE RANGE OF 0 - 15 AND TRUNCATING.

       INTEGER IFLNM(7),OFLNM(7)
       INTEGER MAIN(7),F3(7),MS(2),S1(2),S2(2),S3(2)
       INTEGER TEMP(512),OUT(2048),IMAX,JMAX,MAXCOL,MAXROW
       COMPLEX IN(2048)
       REAL RMAG,MAXVAL,MINVAL,RANGE

       CALL IOF(2,MAIN,IFLNM,OFLNM,F3,MS,S1,S2,S3)

       CALL OPEN(1,IFLNM,1,IER)
       CALL CHECK(IER)
       CALL OPEN(2,OFLNM,3,IER)
       CALL CHECK(IER)

       MAXVAL=0.0
       MINVAL=1E50

       DO 3 I=0,31        ; FIND MAX AND MIN

          CALL RDBLK(1,32*I,IN,32,IER)
          CALL CHECK(IER)

          DO 3 J=1,2048

             RMAG=CABS(IN(J))

             IF(RMAG.LE.MAXVAL)GO TO 2
                MAXVAL=RMAG
                IMAX=I
                JMAX=J

     2       IF(RMAG.LT.MINVAL)MINVAL=RMAG

     3 CONTINUE

       RANGE=MAXVAL-MINVAL

       MAXCOL=MOD(JMAX-1,256)+1
       MAXROW=8*IMAX+INT((JMAX-1)/256)+1

       TYPE"MAX COL = ",MAXCOL     ; REPORT MAX IMFO.
       TYPE"MAX ROW = ",MAXROW

       TYPE"MAX VAL = ",MAXVAL
       TYPE"MIN VAL = ",MINVAL

       DO 5 I=0,31

          CALL RDBLK(1,32*I,IN,32,IER)
          CALL CHECK(IER)

          DO 4 J=1,2048
             OUT(J)=ANINT(15.0*(CABS(IN(J))-MINVAL)/RANGE)
     4    CONTINUE

         CALL REPACK(512,OUT,TEMP)
         CALL WRBLK(2,2*I,TEMP,2,IER)
         CALL CHECK(IER)

     5 CONTINUE

      CALL RESET
      STOP"<7><7><7><7>CPTOVD"
      END
```

AD-A138 421 IMAGE FILTERING WITH BOOLEAN AND STATISTICAL OPERATORS (U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING R D WELLS DEC 83 2/3

UNCLASSIFIED AFIT/GE/EE/83D-72 F/G 20/6 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

APPENDIX B

EDGING PROGRAMS

```
C     PROGRAM STATISTICS - DG FORTRAN 5 - CAPT ROBERT WELLS, DEC 1983

C     THIS PROGRAM COMPUTES THE MEAN AND THE MEAN DEVIATION FOR A
C     NEIGHBORHOOD AROUND EACH PIXEL OF THE INPUT FILE (PACKED VIDEO
C     FORMAT).  THE NEIGHBORHOOD SIZE AND SHAPE ARE USER SELECTABLE.
C     THE PROGRAM USES A FAST ALGORITHM FOR COMPUTING THE MEAN.

      INTEGER OFLNM(7),AFLNM(7),DFLNM(7)
      INTEGER TEMP(512),ORIG(256,32),AVG(256,8),DEV(256,8)
      INTEGER MASKSZ,SHAPE,XOFF,YOFF,PIXROW,BUFROW,COL,ROW
      INTEGER MSUM,FCOLSM,NRMSUM,CHANGE
      REAL MSKCNT

      ACCEPT"ENTER ORIGINAL FILENAME ->"  ; GET FILENAMES
      READ(11,1)OFLNM(1)
    1 FORMAT(S13)
      ACCEPT"ENTER AVERAGE FILENAME ->"
      READ(11,1)AFLNM(1)
      ACCEPT"ENTER DEVIATION FILENAME ->"
      READ(11,1)DFLNM(1)

    2 ACCEPT"ENTER MASK SIZE ->",MASKSZ ;   GET MASKSIZE AND SHAPE
      IF((MASKSZ.NE.3).AND.(MASKSZ.NE.5).AND.(MASKSZ.NE.7).AND.
     * (MASKSZ.NE.11).AND.(MASKSZ.NE.17).AND.(MASKSZ.NE.25))GO TO 2
    3 ACCEPT"CHOOSE MASK SHAPE: 0-HOR, 1-VERT, 2-SQUARE ->",SHAPE
      IF((SHAPE.NE.0).AND.(SHAPE.NE.1).AND.(SHAPE.NE.2))GO TO 3

      CALL OPEN(1,OFLNM,1,IER)    ; OPEN FILES
      CALL CHECK(IER)
      CALL OPEN(2,AFLNM,3,IER)
      CALL CHECK(IER)
      CALL OPEN(3,DFLNM,3,IER)
      CALL CHECK(IER)

      IF(SHAPE.EQ.0)GO TO 4       ; FIND X & Y OFFSETS
      IF(SHAPE.EQ.1)GO TO 5
      XOFF=(MASKSZ-1)/2
      YOFF=XOFF
      GO TO 6
    4 XOFF=(MASKSZ-1)/2
      YOFF=0
      GO TO 6
    5 XOFF=0
      YOFF=(MASKSZ-1)/2
    6 CONTINUE

      MSKCNT=(2*XOFF+1)*(2*YOFF+1) ; NO. OF PIXELS IN MASK

      DO 7 I=1,512  ;FILL IN AVG FILE BORDERS, TOP & BOTTOM
         TEMP(I)=65535
    7 CONTINUE

      CALL WRBLK(2,0,TEMP,2,IER)
      CALL CHECK(IER)
      CALL WRBLK(2,2,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(2,61,TEMP,2,IER)
      CALL CHECK(IER)
      CALL WRBLK(2,63,TEMP,1,IER)
      CALL CHECK(IER)
```

```
      DO 8 I=1,512  ;FILL IN DEV FILE BORDERS, TOP & BOTTOM
         TEMP(I)=0
    8 CONTINUE

      CALL WRBLK(3,0,TEMP,2,IER)
      CALL CHECK(IER)
      CALL WRBLK(3,2,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(3,61,TEMP,2,IER)
      CALL CHECK(IER)
      CALL WRBLK(3,63,TEMP,1,IER)
      CALL CHECK(IER)

      DO 9 I=0,3   ;READ IN FIRST 8 BLOCKS OF ORIG
         CALL RDBLK(1,2*I,TEMP,2,IER)
         CALL CHECK(IER)
         ROW=8*I+1
         CALL UNPACK(512,TEMP,ORIG(1,ROW))
    9 CONTINUE

      MSUM=0       ; COMPUTE FIRST AVERAGING VARIABLES
      DO 10 J=0,2*YOFF
         ROW=13-YOFF+J
         DO 10 K=0,2*XOFF
            COL=13-XOFF+K
            MSUM=MSUM+ORIG(COL,ROW)
   10 CONTINUE

      CHANGE=0
      ROW=13+YOFF
      DO 11 K=0,2*XOFF
         COL=13-XOFF+K
         CHANGE=CHANGE+ORIG(COL,ROW)
   11 CONTINUE

      NRMSUM=MSUM-CHANGE

      DO 13 I=1,28 ; COUNT OF BUFFER LOADS

         DO 12 BUFROW=1,8 ; ROW COUNT WITHIN 2K BUFFERS

            PIXROW=8*I+BUFROW+4
            CALL DOROW(ORIG,AVG,DEV,PIXROW,BUFROW,XOFF,YOFF,MSKCNT,
     *                 MSUM,NRMSUM)

         TYPE"ROW NUMBER:",PIXROW

   12    CONTINUE

         CALL REPACK(512,AVG,TEMP)   ; WRITE 2K BUFFERS OF RESULTS
         CALL WRBLK(2,2*I+1,TEMP,2,IER)
         CALL CHECK(IER)
         CALL REPACK(512,DEV,TEMP)
         CALL WRBLK(3,2*I+1,TEMP,2,IER)
         CALL CHECK(IER)

         CALL RDBLK(1,2*I+6,TEMP,2,IER) ; READ IN NEXT 2K OF ORIG
         CALL CHECK(IER)
         ROW=MOD(8*I+24,32)+1
```

```
      CALL UNPACK(512,TEMP,ORIG(1,ROW))

13 CONTINUE

   DO 14 BUFROW=1,8       ;DO LAST 2K BUFFER

      PIXROW=236+BUFROW
      CALL DOROW(ORIG,AVG,DEV,PIXROW,BUFROW,XOFF,YOFF,MSKCNT,
  *             MSUM,NRMSUM)
      TYPE"ROW NUMBER: ",PIXROW

14 CONTINUE

   CALL REPACK(512,AVG,TEMP)
   CALL WRBLK(2,59,TEMP,2,IER)
   CALL CHECK(IER)
   CALL REPACK(512,DEV,TEMP)
   CALL WRBLK(3,59,TEMP,2,IER)
   CALL CHECK(IER)

   CALL RESET
   STOP STATS
   END
```

```
      SUBROUTINE DOROW(ORIG, AVG, DEV, PIXROW, BUFROW, XOFF, YOFF, MSKCNT,
     *                  MSUM, NRMSUM)

      INTEGER TEMP(512), ORIG(256, 32), AVG(256, 8), DEV(256, 8)
      INTEGER XOFF, YOFF, PIXROW, BUFROW, ROW, COL
      INTEGER MSUM, FCOLSM, NRMSUM, CHANGE, PIXCOL, DSUM
      REAL MSKCNT

      DO 20 K=1, 12 ; FILL IN FIRST 12 PIXELS
         AVG(K, BUFROW)=15
         DEV(K, BUFROW)=0
   20 CONTINUE

      DO 21 K=245, 256 ; FILL IN LAST 12 PIXELS
         AVG(K, BUFROW)=15
         DEV(K, BUFROW)=0
   21 CONTINUE

      CHANGE=0
      ROW=MOD(PIXROW+YOFF-1, 32)+1
      DO 22 K=0, 2*XOFF
         COL=13-XOFF+K
         CHANGE=CHANGE+ORIG(COL, ROW)
   22 CONTINUE

      MSUM=NRMSUM+CHANGE      ; NEW MATRIX SUM

      CHANGE=0
      ROW=MOD(PIXROW-YOFF-1, 32)+1
      DO 23 K=0, 2*XOFF
         COL=13-XOFF+K
         CHANGE=CHANGE+ORIG(COL, ROW)
   23 CONTINUE

      NRMSUM=MSUM-CHANGE      ; NEW NEXT. ROW. MSUM

      FCOLSM=0           ; COMPUTE NEW FIRST COL SUM
      COL=13-XOFF
      DO 24 J=0, 2*YOFF
         ROW=MOD(PIXROW-YOFF+J-1, 32)+1
         FCOLSM=FCOLSM+ORIG(COL, ROW)
   24 CONTINUE

      AVG(13, BUFROW)=ANINT(MSUM/MSKCNT) ; PIXEL 13 AVERAGE

      DO 26 PIXCOL=14, 244 ; AVG ROUTINE FOR PIXELS 14-244

         CHANGE=0
         COL=PIXCOL+XOFF
         DO 25 J=0, 2*YOFF
            ROW=MOD(PIXROW-YOFF+J-1, 32)+1
            CHANGE=CHANGE+ORIG(COL, ROW)
   25    CONTINUE

         MSUM=MSUM+CHANGE-FCOLSM

         AVG(PIXCOL, BUFROW)=ANINT(MSUM/MSKCNT)

         FCOLSM=0
         COL=PIXCOL-XOFF
```

```
      DO 26 J=0,2*YOFF
         ROW=MOD(PIXROW-YOFF+J-1,32)+1
         FCOLSM=FCOLSM+ORIG(COL,ROW)

26 CONTINUE

   DO 28 PIXCOL=13,244 ; DEVIATION ROUTINE

   DSUM=0
   DO 27 J=0,2*YOFF
      ROW=MOD(PIXROW-YOFF+J-1,32)+1
      DO 27 K=0,2*XOFF
         COL=PIXCOL-XOFF+K
         DSUM=DSUM+ABS(ORIG(COL,ROW)-AVG(PIXCOL,BUFROW))
27    CONTINUE

      DEV(PIXCOL,BUFROW)=ANINT(4*DSUM/MSKCNT)

      IF(DEV(PIXCOL,BUFROW).GT.15)DEV(PIXCOL,BUFROW)=15

28 CONTINUE

   RETURN
   END
```

```
C     PROGRAM KEDGE - DG FORTRAN 5 - CAPT ROBERT WELLS, DEC 1983

C     THIS PROGRAM EDGES THE INPUT FILE (PACKED VIDEO FORMAT) WITH THE
C     KIRSCH OPERATOR. THE COEFFICIENTS ARE NOT 5 AND -3 BUT THE RATIO
C     IS 5 TO -3. THE COEFFICIENTS WERE REDUCED TO KEEP THE EDGE OUTPUT
C     IN THE 0 - 15 RANGE. THE OUTPUT IS 256 X 256 PACKED VIDEO DATA.

      INTEGER INMAT(256,32),OUTMAT(256,16),TEMP(1024),TOGGLE
      INTEGER IFLNM(7),OFLNM(7)
      INTEGER MAIN(7),F3(7),MS(2),S1(2),S2(2),S3(2)

      CALL IOF(2,MAIN,IFLNM,OFLNM,F3,MS,S1,S2,S3)

      CALL OPEN(1,IFLNM,1,IER)
      CALL CHECK(IER)
      CALL OPEN(2,OFLNM,3,IER)
      CALL CHECK(IER)

      DO 1 I=1,512     ; FILL FIRST AND LAST 8 ROWS OF
         TEMP(I)=0     ; OF RESULTS WITH ZERO
    1 CONTINUE
      CALL WRBLK(2,0,TEMP,2,IER)
      CALL CHECK(IER)
      CALL WRBLK(2,62,TEMP,2,IER)
      CALL CHECK(IER)

      CALL RDBLK(1,0,TEMP,4,IER) ;READ IN FIRST 16 ROWS
      CALL CHECK(IER)
      CALL UNPACK(1024,TEMP,INMAT)

      TOGGLE=0   ; FLAG TO SHOW BUFFER WRAP-AROUND

      DO 5 I=1,15 ; COUNTER FOR 4K BUFFER LOADS

         CALL RDBLK(1,4*I,TEMP,4,IER) ;READ NEXT 16 ROWS
         CALL CHECK(IER)
         CALL UNPACK(1024,TEMP,INMAT(1,17-16*TOGGLE))

         DO 3 K=1,16 ; COUNTER FOR OUTPUT BUFFER ROWS

            DO 2 J=1,8 ; ZERO FILL FIRST/LAST 8 COLUMNS
               OUTMAT(J,K)=0
               OUTMAT(J+248,K)=0
    2       CONTINUE

            DO 3 J=9,248 ; EDGE THIS ROW
               CALL EDGER3(J,K,TOGGLE,INMAT,OUTMAT)
    3    CONTINUE

         CALL REPACK(1024,OUTMAT,TEMP)
         CALL WRBLK(2,4*I-2,TEMP,4,IER) ; WRITE 16 ROWS OF RESULTS
         CALL CHECK(IER)

         IF(TOGGLE.EQ.1)GO TO 4
            TOGGLE=1     ; TOGGLE THE BUFFER WRAP-AROUND FLAG
            GO TO 5
    4       TOGGLE=0

    5 CONTINUE

      CALL RESET
      STOP"<7><7><7><7>KEDGE"
      END
```

```
      SUBROUTINE EDGER3(J,K,TOGGLE,INMAT,OUTMAT)

      INTEGER J,K,TOGGLE,INMAT(256,32),OUTMAT(256,16)
      INTEGER J1,J2,J3,K1,K2,K3
      REAL A,B,C,D,E,F,G,H,MAXIM

      IF(TOGGLE.EQ.1)GO TO 1
         K1=K+7
         K2=K+8
         K3=K+9
         GO TO 2
1        K1=MOD(K+22,32)+1
         K2=MOD(K+23,32)+1
         K3=MOD(K+24,32)+1

2     J1=J-1
      J2=J
      J3=J+1

      A=0.667*(INMAT(J1,K1)+INMAT(J2,K1)+INMAT(J3,K1))-
     +  0.400*(INMAT(J3,K2)+INMAT(J3,K3)+INMAT(J2,K3)+
     +         INMAT(J1,K3)+INMAT(J1,K2))

      B=0.667*(INMAT(J2,K1)+INMAT(J3,K1)+INMAT(J3,K2))-
     +  0.400*(INMAT(J3,K3)+INMAT(J2,K3)+INMAT(J1,K3)+
     +         INMAT(J1,K2)+INMAT(J1,K1))

      C=0.667*(INMAT(J3,K1)+INMAT(J3,K2)+INMAT(J3,K3))-
     +  0.400*(INMAT(J2,K3)+INMAT(J1,K3)+INMAT(J1,K2)+
     +         INMAT(J1,K1)+INMAT(J2,K1))

      D=0.667*(INMAT(J3,K2)+INMAT(J3,K3)+INMAT(J2,K3))-
     +  0.400*(INMAT(J1,K3)+INMAT(J1,K2)+INMAT(J1,K1)+
     +         INMAT(J2,K1)+INMAT(J3,K1))

      E=0.667*(INMAT(J3,K3)+INMAT(J2,K3)+INMAT(J1,K3))-
     +  0.400*(INMAT(J1,K2)+INMAT(J1,K1)+INMAT(J2,K1)+
     +         INMAT(J3,K1)+INMAT(J3,K2))

      F=0.667*(INMAT(J2,K3)+INMAT(J1,K3)+INMAT(J1,K2))-
     +  0.400*(INMAT(J1,K1)+INMAT(J2,K1)+INMAT(J3,K1)+
     +         INMAT(J3,K2)+INMAT(J3,K3))

      G=0.667*(INMAT(J1,K3)+INMAT(J1,K2)+INMAT(J1,K1))-
     +  0.400*(INMAT(J2,K1)+INMAT(J3,K1)+INMAT(J3,K2)+
     +         INMAT(J3,K3)+INMAT(J2,K3))

      H=0.667*(INMAT(J1,K2)+INMAT(J1,K1)+INMAT(J2,K1))-
     +  0.400*(INMAT(J3,K1)+INMAT(J3,K2)+INMAT(J3,K3)+
     +         INMAT(J2,K3)+INMAT(J1,K3))

      MAXIM=AMAX1(A,B,C,D,E,F,G,H)

      OUTMAT(J,K)=ANINT(MAXIM)

      IF(OUTMAT(J,K).GT.15)OUTMAT(J,K)=15   ; CHECK FOR CLIPPING

      RETURN
      END
```

```
C     PROGRAM WEDGE - DG FORTRAN 5 - CAPT ROBERT WELLS, DEC 1983

C     THIS PROGRAM EDGES THE INPUT FILE (PACKED VIDEO FORMAT) WITH
C     A MASK OPERATOR CALLED THE WEDGE OPERATOR.  THE OUTPUT IS SEPARATED
C     INTO FOUR 256 X 256 PACKED VIDEO FILES ACCORDING TO WHICH MASK
C     PRODUCED THE MAXIMUM VALUE.  IF TWO MASKS OF DIFFERENT ORIENTATIONS
C     PRODUCE THE MAXIMUM, THAT VALUE GOES TO BOTH OUTPUT FILES.

      INTEGER INMAT(256,16),TEMP(512),TOGGLE
      INTEGER OMAT1(256,8),OMAT2(256,8),OMAT3(256,8),OMAT4(256,8)
      INTEGER INFLNM(7),OFLNM1(7),OFLNM2(7),OFLNM3(7),OFLNM4(7)

      ACCEPT"ENTER INPUT FILENAME ->"
      READ(11,1)INFLNM(1)
    1 FORMAT(S13)
      ACCEPT"ENTER OUTPUT FILENAME #1 ->"
      READ(11,1)OFLNM1(1)
      ACCEPT"ENTER OUTPUT FILENAME #2 ->"
      READ(11,1)OFLNM2(1)
      ACCEPT"ENTER OUTPUT FILENAME #3 ->"
      READ(11,1)OFLNM3(1)
      ACCEPT"ENTER OUTPUT FILENAME #4 ->"
      READ(11,1)OFLNM4(1)

      CALL OPEN(1,INFLNM,1,IER)
      CALL CHECK(IER)
      CALL OPEN(2,OFLNM1,3,IER)
      CALL CHECK(IER)
      CALL OPEN(3,OFLNM2,3,IER)
      CALL CHECK(IER)
      CALL OPEN(4,OFLNM3,3,IER)
      CALL CHECK(IER)
      CALL OPEN(5,OFLNM4,3,IER)
      CALL CHECK(IER)

      DO 2 I=1,256      ; FILL FIRST AND LAST 4 ROWS OF
         TEMP(I)=0      ; OF OUTPUT FILES WITH ZERO
    2 CONTINUE
      CALL WRBLK(2,0,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(2,63,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(3,0,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(3,63,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(4,0,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(4,63,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(5,0,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(5,63,TEMP,1,IER)
      CALL CHECK(IER)

      CALL RDBLK(1,0,TEMP,2,IER) ;READ IN FIRST 8 ROWS
      CALL CHECK(IER)
      CALL UNPACK(512,TEMP,INMAT)

      TOGGLE=0   ; FLAG TO SHOW BUFFER WRAP-AROUND
```

```
      DO 5 I=1,31 ; COUNTER FOR 2K BUFFER LOADS

         CALL RDBLK(1,2*I,TEMP,2,IER) ;READ NEXT 8 ROWS
         CALL CHECK(IER)
         CALL UNPACK(512,TEMP,INMAT(1,9-8*TOGGLE))

         DO 3 K=1,8 ; COUNTER FOR OUTPUT BUFFER ROWS
              CALL WROW1(K,TOGGLE,INMAT,OMAT1,OMAT2,OMAT3,OMAT4)
3        CONTINUE

         CALL REPACK(512,OMAT1,TEMP)
         CALL WRBLK(2,2*I-1,TEMP,2,IER) ; WRITE 8 ROWS OF RESULTS
         CALL CHECK(IER)
         CALL REPACK(512,OMAT2,TEMP)
         CALL WRBLK(3,2*I-1,TEMP,2,IER)
         CALL CHECK(IER)
         CALL REPACK(512,OMAT3,TEMP)
         CALL WRBLK(4,2*I-1,TEMP,2,IER)
         CALL CHECK(IER)
         CALL REPACK(512,OMAT4,TEMP)
         CALL WRBLK(5,2*I-1,TEMP,2,IER)
         CALL CHECK(IER)

         IF(TOGGLE.EQ.1)GO TO 4
            TOGGLE=1     ; TOGGLE THE BUFFER WRAP-AROUND FLAG
            GO TO 5
4           TOGGLE=0

5     CONTINUE

      CALL RESET
      STOP"<7><7><7><7>WEDGE"
      END
```

```
      SUBROUTINE WROW1(K,TOGGLE,INMAT,OMAT1,OMAT2,OMAT3,OMAT4)

      INTEGER K,TOGGLE,INMAT(256,16),MEDGE
      INTEGER OMAT1(256,8),OMAT2(256,8),OMAT3(256,8),OMAT4(256,8)
      REAL A1,A2,A3,A4,B1,B2,B3,B4,C1,C2,C3,C4,D1,D2,D3,D4,MAXIM

      IF(TOGGLE.EQ.1)GO TO 1   ; FIND INPUT BUFFER INDEXES
         K1=K+3
         K2=K+4
         K3=K+5
         GO TO 2
1        K1=MOD(K+10,16)+1
         K2=MOD(K+11,16)+1
         K3=MOD(K+12,16)+1
2     CONTINUE

      DO 3 J=1,4      ; ZERO FILL FIRST/LAST 4 COLUMNS OF RESULTS
         OMAT1(J,K)=0
         OMAT1(J+252,K)=0
         OMAT2(J,K)=0
         OMAT2(J+252,K)=0
         OMAT3(J,K)=0
         OMAT3(J+252,K)=0
         OMAT4(J,K)=0
         OMAT4(J+252,K)=0
3     CONTINUE

      DO 4 J=5,252   ; EDGE THESE COLUMNS

         J1=J-1
         J2=J
         J3=J+1

         A1=0.600*(INMAT(J1,K1)+INMAT(J2,K1)+INMAT(J3,K1))-
     +      0.300*(INMAT(J1,K2)+INMAT(J2,K2)+INMAT(J3,K2)+
     +             INMAT(J1,K3)+INMAT(J2,K3)+INMAT(J3,K3))

         A2=0.600*(INMAT(J1,K3)+INMAT(J2,K3)+INMAT(J3,K3))-
     +      0.300*(INMAT(J1,K2)+INMAT(J2,K2)+INMAT(J3,K2)+
     +             INMAT(J1,K1)+INMAT(J2,K1)+INMAT(J3,K1))

         A3=-A1

         A4=-A2

         B1=0.600*(INMAT(J2,K1)+INMAT(J3,K1)+INMAT(J3,K2))-
     +      0.300*(INMAT(J1,K1)+INMAT(J2,K2)+INMAT(J3,K3)+
     +             INMAT(J1,K2)+INMAT(J1,K3)+INMAT(J2,K3))

         B2=0.600*(INMAT(J1,K2)+INMAT(J1,K3)+INMAT(J2,K3))-
     +      0.300*(INMAT(J1,K1)+INMAT(J2,K2)+INMAT(J3,K3)+
     +             INMAT(J2,K1)+INMAT(J3,K1)+INMAT(J3,K2))

         B3=-B1

         B4=-B2

         C1=0.600*(INMAT(J3,K1)+INMAT(J3,K2)+INMAT(J3,K3))-
     +      0.300*(INMAT(J2,K1)+INMAT(J2,K2)+INMAT(J2,K3)+
     +             INMAT(J1,K1)+INMAT(J1,K2)+INMAT(J1,K3))
```

```
      C2=0.600*(INMAT(J1,K1)+INMAT(J1,K2)+INMAT(J1,K3))-
     +   0.300*(INMAT(J2,K1)+INMAT(J2,K2)+INMAT(J2,K3)+
     +          INMAT(J3,K1)+INMAT(J3,K2)+INMAT(J3,K3))

      C3=-C1

      C4=-C2

      D1=0.600*(INMAT(J1,K2)+INMAT(J1,K1)+INMAT(J2,K1))-
     +   0.300*(INMAT(J1,K3)+INMAT(J2,K2)+INMAT(J3,K1)+
     +          INMAT(J2,K3)+INMAT(J3,K3)+INMAT(J3,K2))

      D2=0.600*(INMAT(J2,K3)+INMAT(J3,K3)+INMAT(J3,K2))-
     +   0.300*(INMAT(J1,K3)+INMAT(J2,K2)+INMAT(J3,K1)+
     +          INMAT(J1,K2)+INMAT(J1,K1)+INMAT(J2,K1))

      D3=-D1

      D4=-D2

      MAXIM=AMAX1(A1,A2,A3,A4,B1,B2,B3,B4,C1,C2,C3,C4,D1,D2,D3,D4)

      OMAT1(J,K)=0
      OMAT2(J,K)=0
      OMAT3(J,K)=0
      OMAT4(J,K)=0

      MEDGE=ANINT(MAXIM)      ; INTEGRAL EDGE VALUE

      IF(MEDGE.GT.15)MEDGE=15 ; TEST FOR OVERFLOW
      IF(MEDGE.LT.0)MEDGE=0   ; AND UNDERFLOW

      IF(A1.EQ.MAXIM.OR.A2.EQ.MAXIM.OR.A3.EQ.MAXIM.OR.A4.EQ.MAXIM)
     +   OMAT1(J,K)=MEDGE    ; ORIENTATION #1

      IF(B1.EQ.MAXIM.OR.B2.EQ.MAXIM.OR.B3.EQ.MAXIM.OR.B4.EQ.MAXIM)
     +   OMAT2(J,K)=MEDGE    ; ORIENTATION #2

      IF(C1.EQ.MAXIM.OR.C2.EQ.MAXIM.OR.C3.EQ.MAXIM.OR.C4.EQ.MAXIM)
     +   OMAT3(J,K)=MEDGE    ; ORIENTATION #3

      IF(D1.EQ.MAXIM.OR.D2.EQ.MAXIM.OR.D3.EQ.MAXIM.OR.D4.EQ.MAXIM)
     +   OMAT4(J,K)=MEDGE    ; ORIENTATION #4

    4 CONTINUE

      RETURN
      END
```

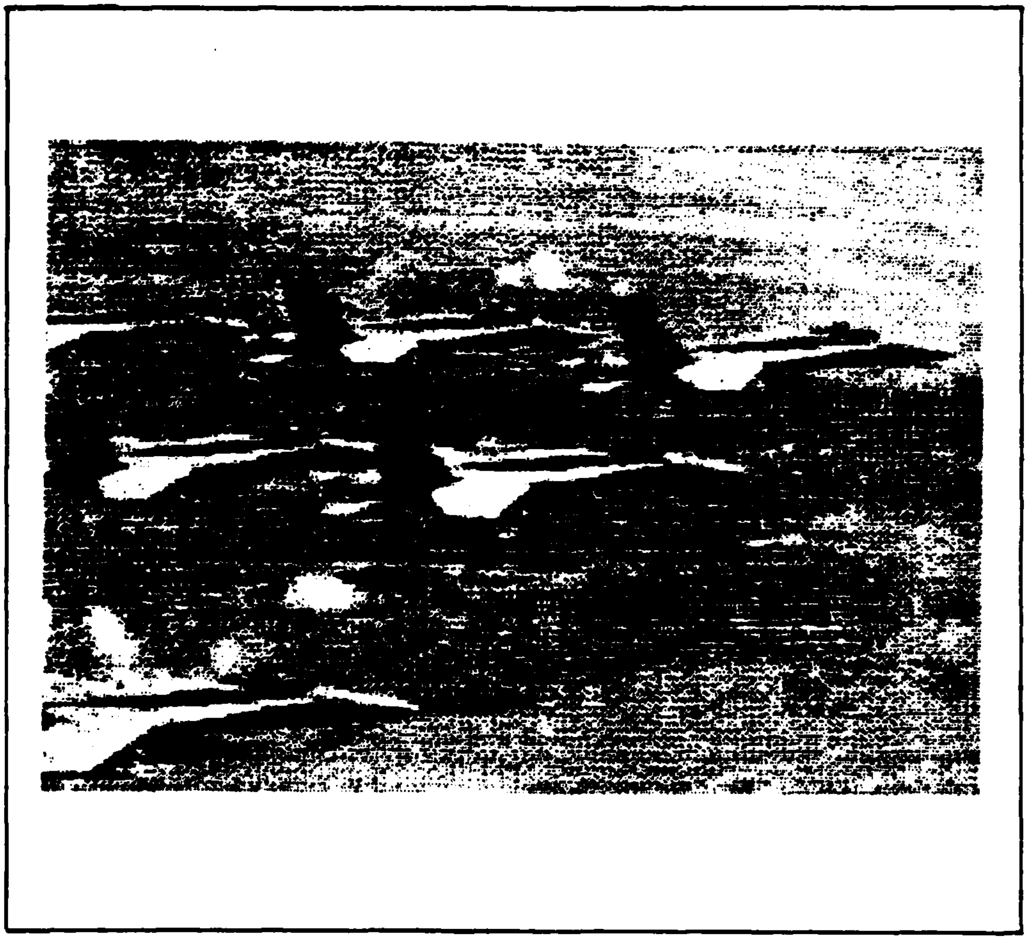
```
C     PROGRAM TEDGE - DG FORTRAN 5 - CAPT ROBERT WELLS, DEC 1983

C     THIS PROGRAM DETECTS LINES IN THE INPUT FILE (PACKED VIDEO FORMAT)
C     WITH A MASK OPERATOR FOR LINE DETECTION AT FOUR ORIENTATIONS.
C     THE OUTPUT IS SEPARATED INTO FOUR 256 X 256 PACKED VIDEO FORMAT
C     FILES ACCORDING TO WHICH MASK PRODUCED THE MAXIMUM VALUE.  IF TWO OR
C     MORE MASKS PRODUCE THE SAME MAXIMUM, THE OUTPUT GOES TO EACH
C     RESPECTIVE FILE.

      INTEGER INMAT(256,16),TEMP(512),TOGGLE
      INTEGER OMAT1(256,8),OMAT2(256,8),OMAT3(256,8),OMAT4(256,8)
      INTEGER INFLNM(7),OFLNM1(7),OFLNM2(7),OFLNM3(7),OFLNM4(7)

      ACCEPT"ENTER INPUT FILENAME ->"
      READ(11,1)INFLNM(1)
    1 FORMAT(S13)
      ACCEPT"ENTER OUTPUT FILENAME #1 ->"
      READ(11,1)OFLNM1(1)
      ACCEPT"ENTER OUTPUT FILENAME #2 ->"
      READ(11,1)OFLNM2(1)
      ACCEPT"ENTER OUTPUT FILENAME #3 ->"
      READ(11,1)OFLNM3(1)
      ACCEPT"ENTER OUTPUT FILENAME #4 ->"
      READ(11,1)OFLNM4(1)

      CALL OPEN(1,INFLNM,1,IER)
      CALL CHECK(IER)
      CALL OPEN(2,OFLNM1,3,IER)
      CALL CHECK(IER)
      CALL OPEN(3,OFLNM2,3,IER)
      CALL CHECK(IER)
      CALL OPEN(4,OFLNM3,3,IER)
      CALL CHECK(IER)
      CALL OPEN(5,OFLNM4,3,IER)
      CALL CHECK(IER)

      DO 2 I=1,256    ; FILL FIRST AND LAST 4 ROWS OF
         TEMP(I)=0    ; OF OUTPUT FILES WITH ZERO
    2 CONTINUE
      CALL WRBLK(2,0,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(2,63,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(3,0,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(3,63,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(4,0,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(4,63,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(5,0,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(5,63,TEMP,1,IER)
      CALL CHECK(IER)

      CALL RDBLK(1,0,TEMP,2,IER) ;READ IN FIRST 8 ROWS
      CALL CHECK(IER)
      CALL UNPACK(512,TEMP,INMAT)
```

```
      TOGGLE=0  ; FLAG TO SHOW BUFFER WRAP-AROUND

      DO 5 I=1,31 ; COUNTER FOR 2K BUFFER LOADS

         CALL RDBLK(1,2*I,TEMP,2,IER) ;READ NEXT 8 ROWS
         CALL CHECK(IER)
         CALL UNPACK(512,TEMP,INMAT(1,9-8*TOGGLE))

         DO 3 K=1,8 ; COUNTER FOR OUTPUT BUFFER ROWS
               CALL TROW1(K,TOGGLE,INMAT,OMAT1,OMAT2,OMAT3,OMAT4)
3        CONTINUE

         CALL REPACK(512,OMAT1,TEMP)
         CALL WRBLK(2,2*I-1,TEMP,2,IER) ; WRITE 8 ROWS OF RESULTS
         CALL CHECK(IER)
         CALL REPACK(512,OMAT2,TEMP)
         CALL WRBLK(3,2*I-1,TEMP,2,IER)
         CALL CHECK(IER)
         CALL REPACK(512,OMAT3,TEMP)
         CALL WRBLK(4,2*I-1,TEMP,2,IER)
         CALL CHECK(IER)
         CALL REPACK(512,OMAT4,TEMP)
         CALL WRBLK(5,2*I-1,TEMP,2,IER)
         CALL CHECK(IER)

         IF(TOGGLE.EQ.1)GO TO 4
            TOGGLE=1     ; TOGGLE THE BUFFER WRAP-AROUND FLAG
            GO TO 5
4           TOGGLE=0

5     CONTINUE

      CALL RESET
      STOP"<7><7><7><7>TEDGE"
      END
```

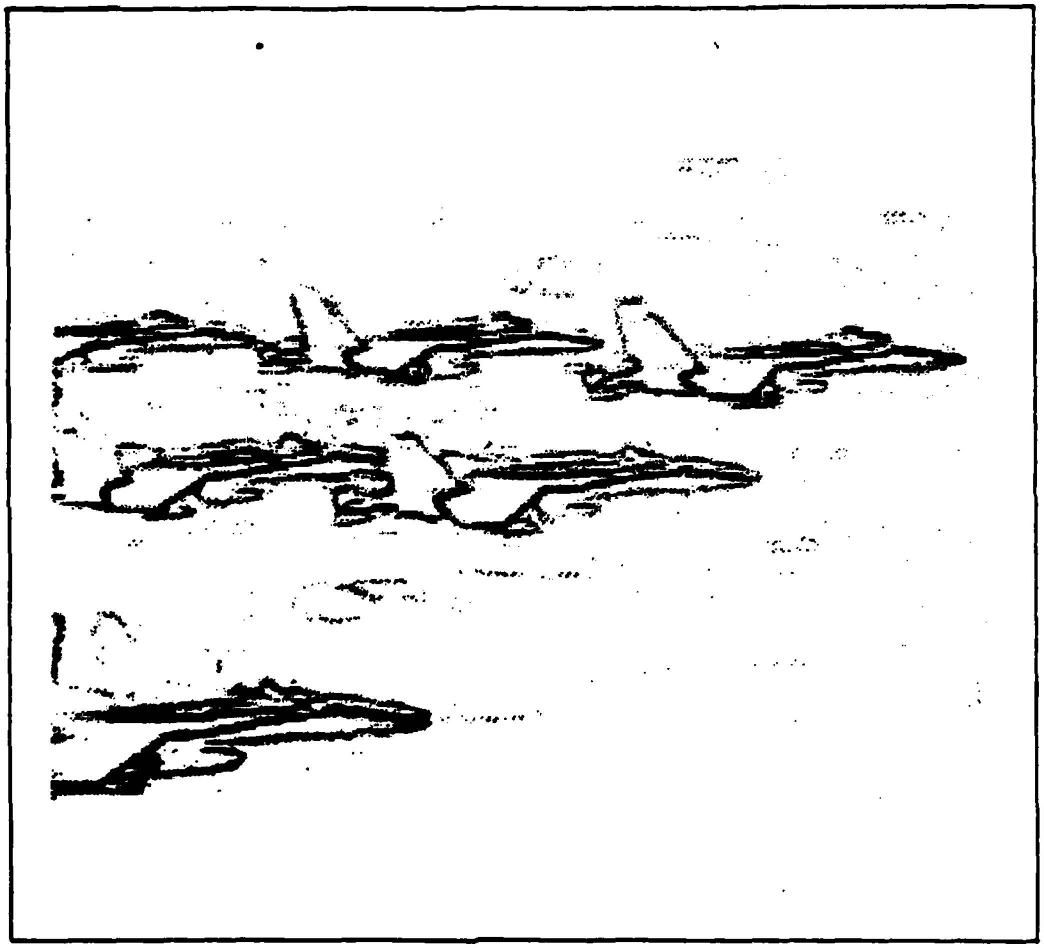
```
      SUBROUTINE TROW1(K,TOGGLE,INMAT,OMAT1,OMAT2,OMAT3,OMAT4)

      INTEGER K,TOGGLE,INMAT(256,16),A,B,C,D,MAXIM
      INTEGER OMAT1(256,8),OMAT2(256,8),OMAT3(256,8),OMAT4(256,8)

      IF(TOGGLE.EQ.1)GO TO 1   ; FIND INPUT BUFFER INDEXES
         K1=K+3
         K2=K+4
         K3=K+5
         GO TO 2
1        K1=MOD(K+10,16)+1
         K2=MOD(K+11,16)+1
         K3=MOD(K+12,16)+1
2     CONTINUE

      DO 3 J=1,4       ; ZERO FILL FIRST/LAST 4 COLUMNS OF RESULTS
         OMAT1(J,K)=0
         OMAT1(J+252,K)=0
         OMAT2(J,K)=0
         OMAT2(J+252,K)=0
         OMAT3(J,K)=0
         OMAT3(J+252,K)=0
         OMAT4(J,K)=0
         OMAT4(J+252,K)=0
3     CONTINUE

      DO 4 J=5,252   ; SEPERATE THE TEMPLATE EDGES

         J1=J-1
         J2=J
         J3=J+1

         OMAT1(J,K)=0
         OMAT2(J,K)=0
         OMAT3(J,K)=0
         OMAT4(J,K)=0

         IF(INMAT(J2,K2).EQ.0)GO TO 4 ; CENTER PIXEL SHOULD NOT EQUAL 0

         A=INMAT(J1,K2)+INMAT(J2,K2)+INMAT(J3,K2)

         B=INMAT(J1,K1)+INMAT(J2,K2)+INMAT(J3,K3)

         C=INMAT(J2,K1)+INMAT(J2,K2)+INMAT(J2,K3)

         D=INMAT(J1,K3)+INMAT(J2,K2)+INMAT(J3,K1)

         MAXIM=MAXO(A,B,C,D)

         IF(MAXIM.LT.20)GO TO 4 ; TEST TO ELIMINATE NOISE

         IF(A.EQ.MAXIM)OMAT1(J,K)=15   ; ASSIGN EDGE VALUE TO
                                       ; PROPER OUTPUT BUFFER(S)
         IF(B.EQ.MAXIM)OMAT2(J,K)=15

         IF(C.EQ.MAXIM)OMAT3(J,K)=15

         IF(D.EQ.MAXIM)OMAT4(J,K)=15

4     CONTINUE

      RETURN
      END
```

APPENDIX C

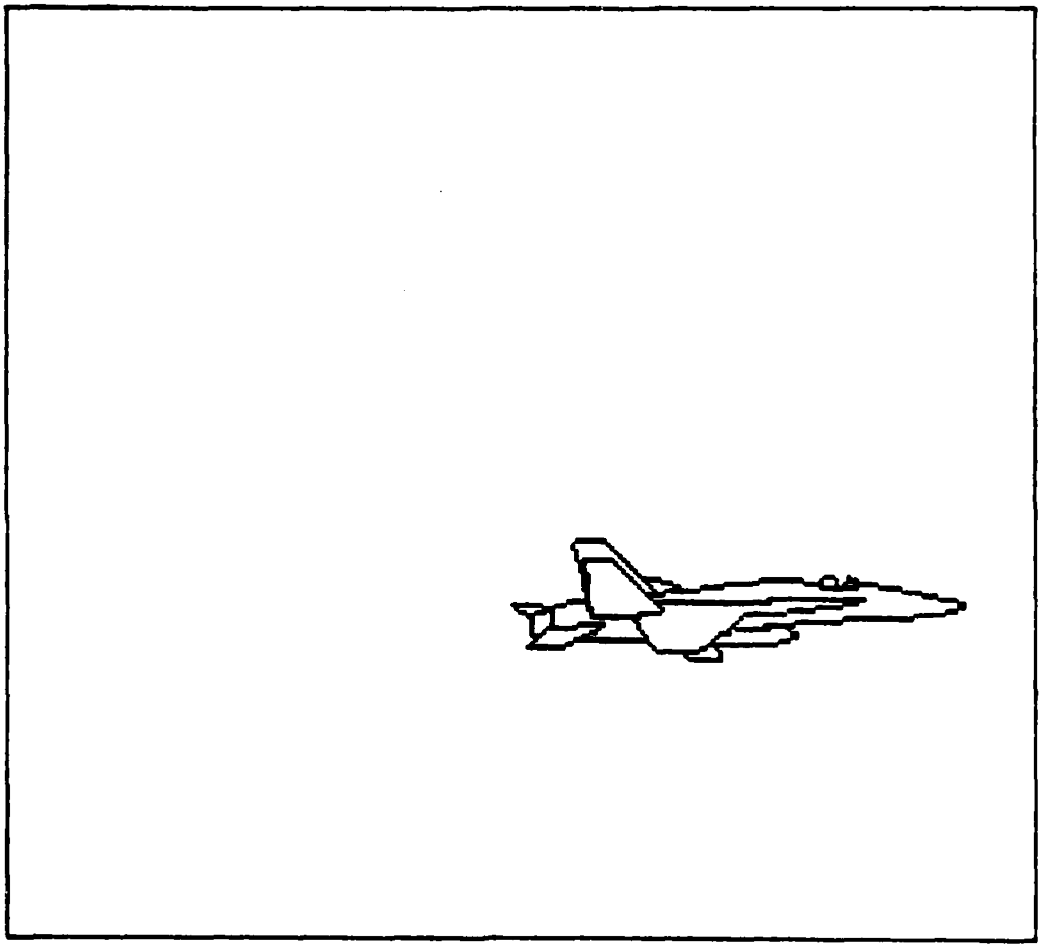
MORE CORRELATION RESULTS



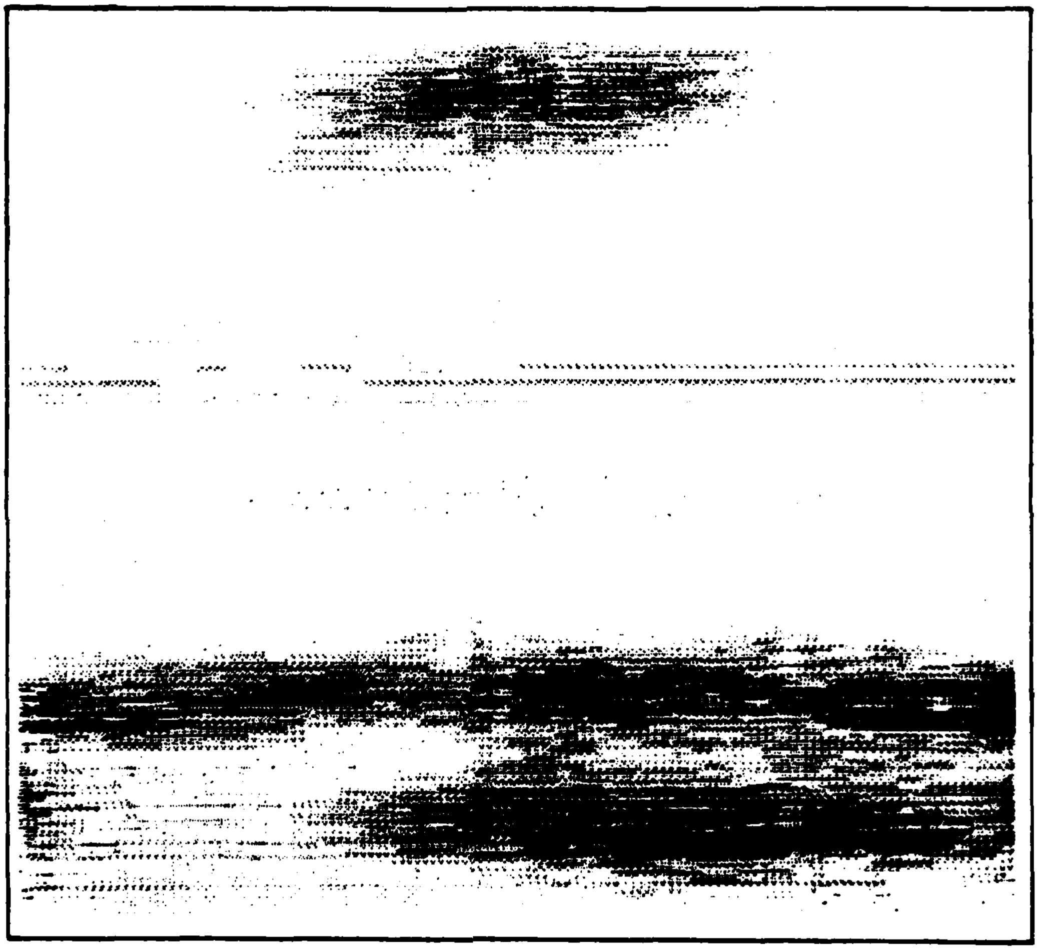
Original F18 Hornet



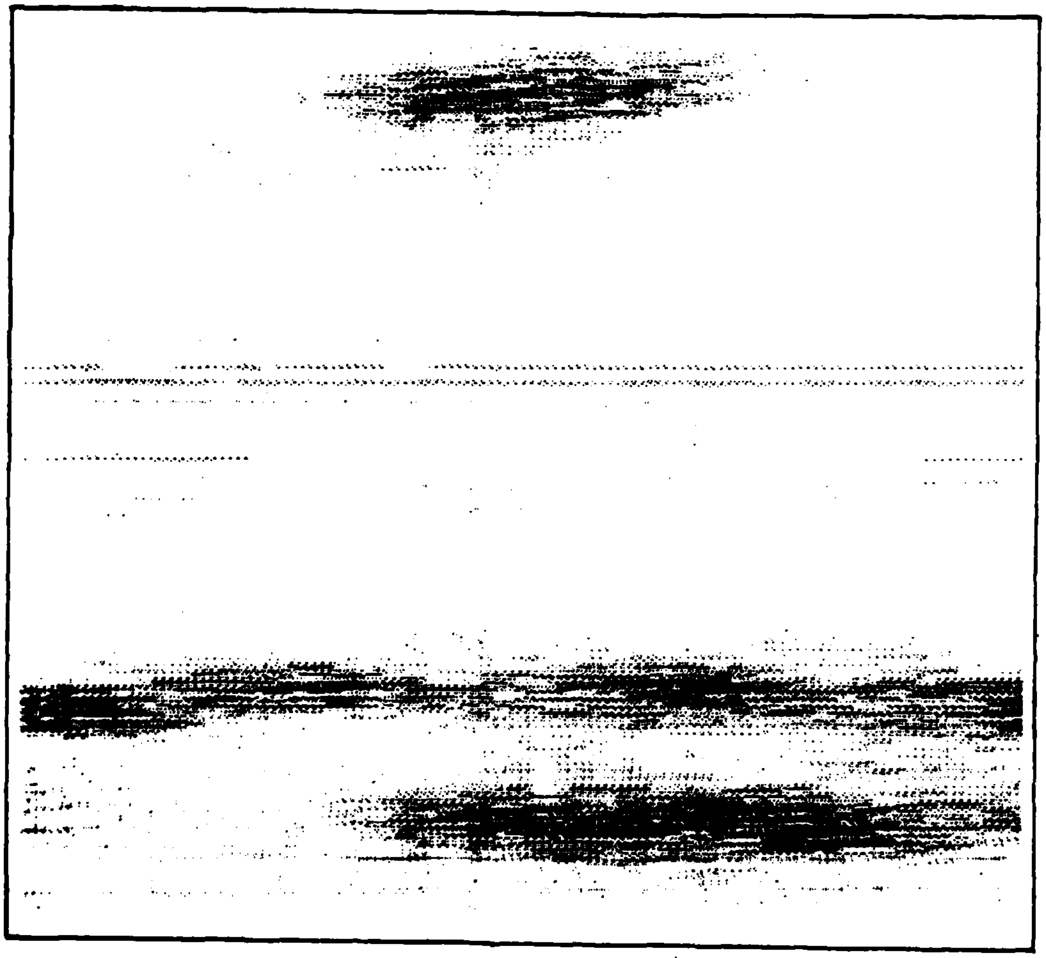
Combined Result of Wedge Operator on F18



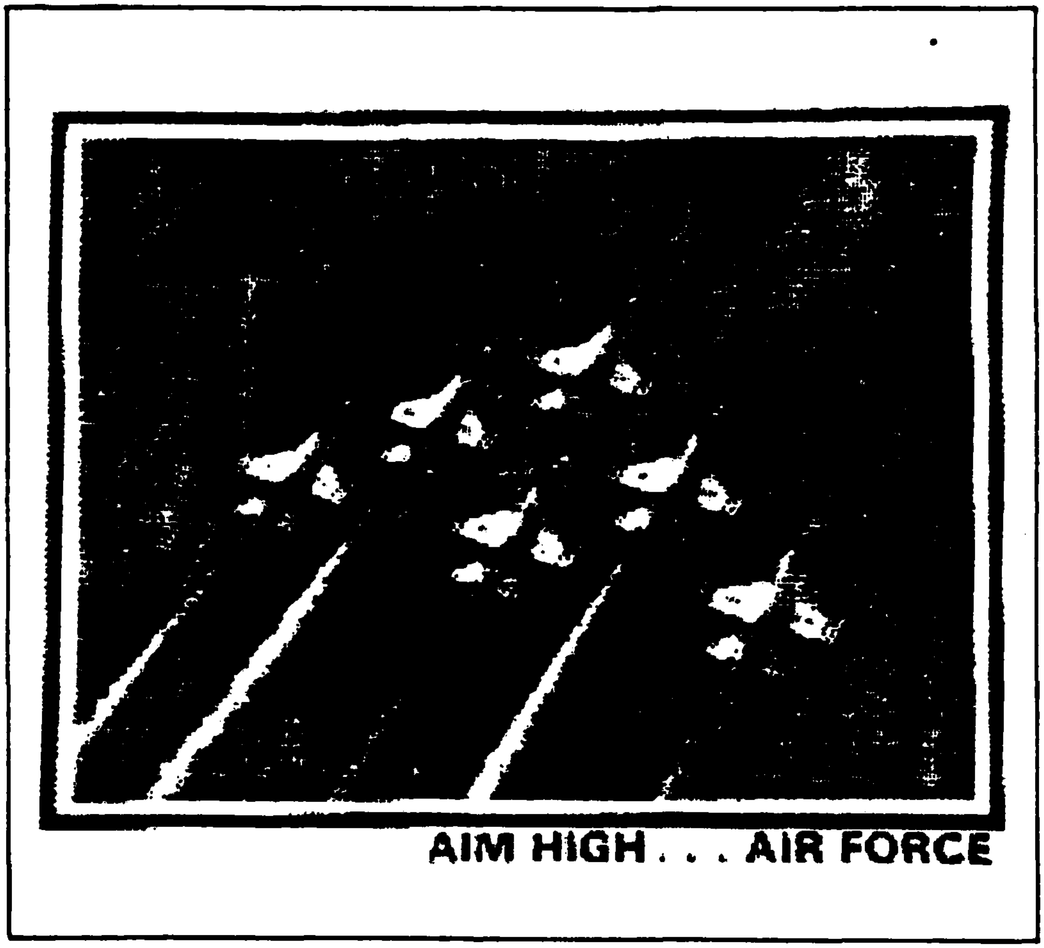
F18 Template



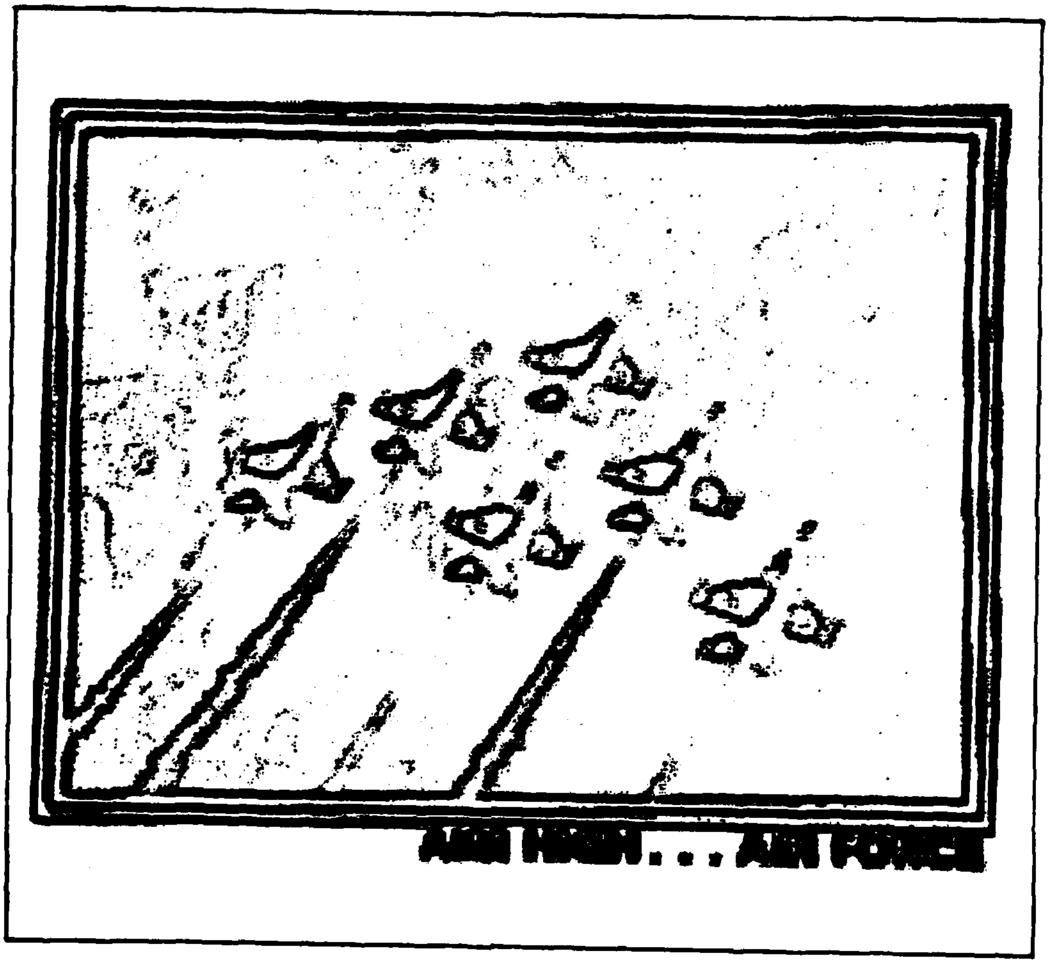
Correlation of F18



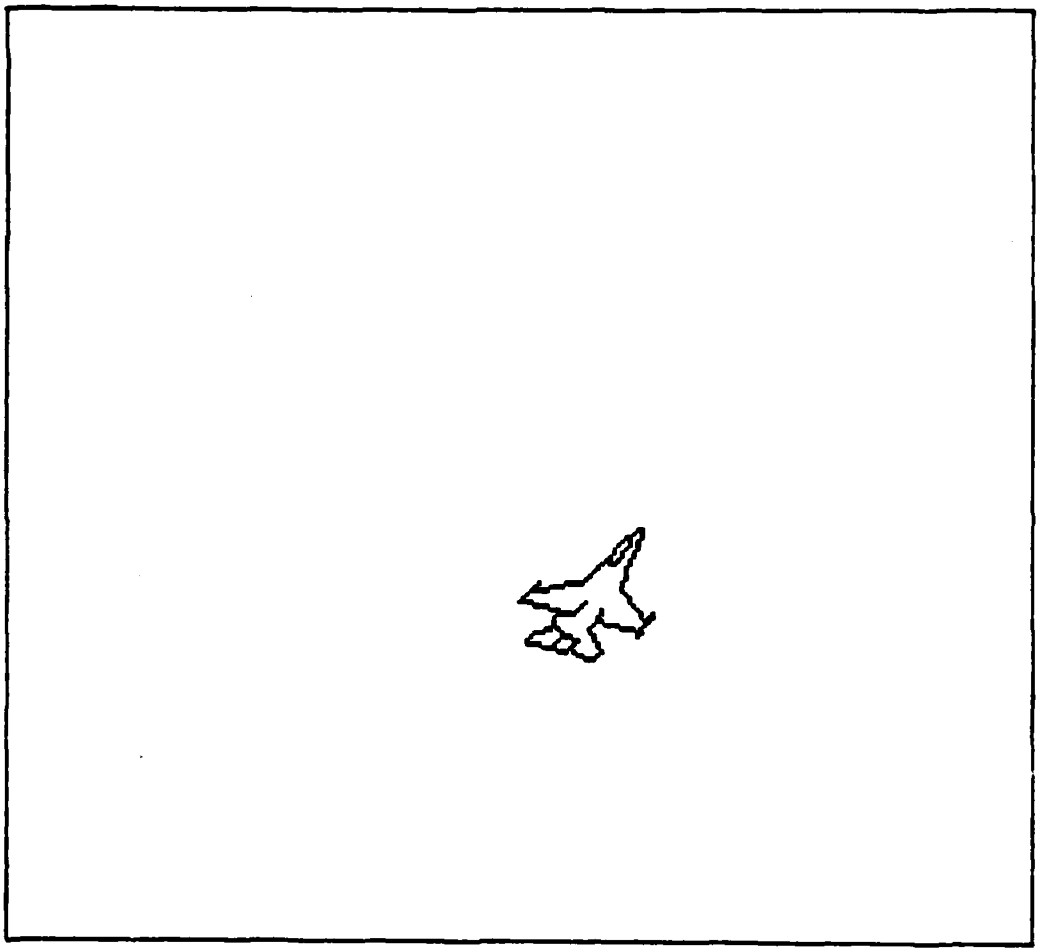
Separated Orientation Correlation of F18



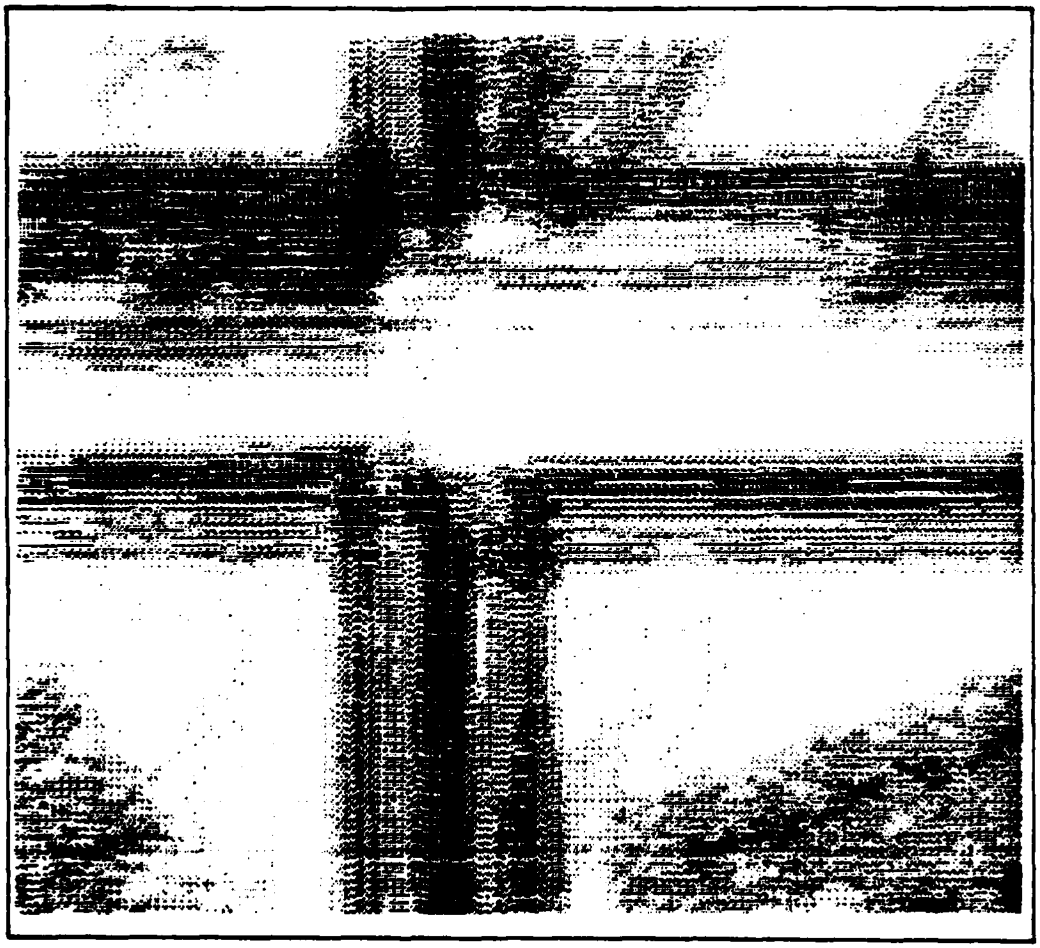
Original F16



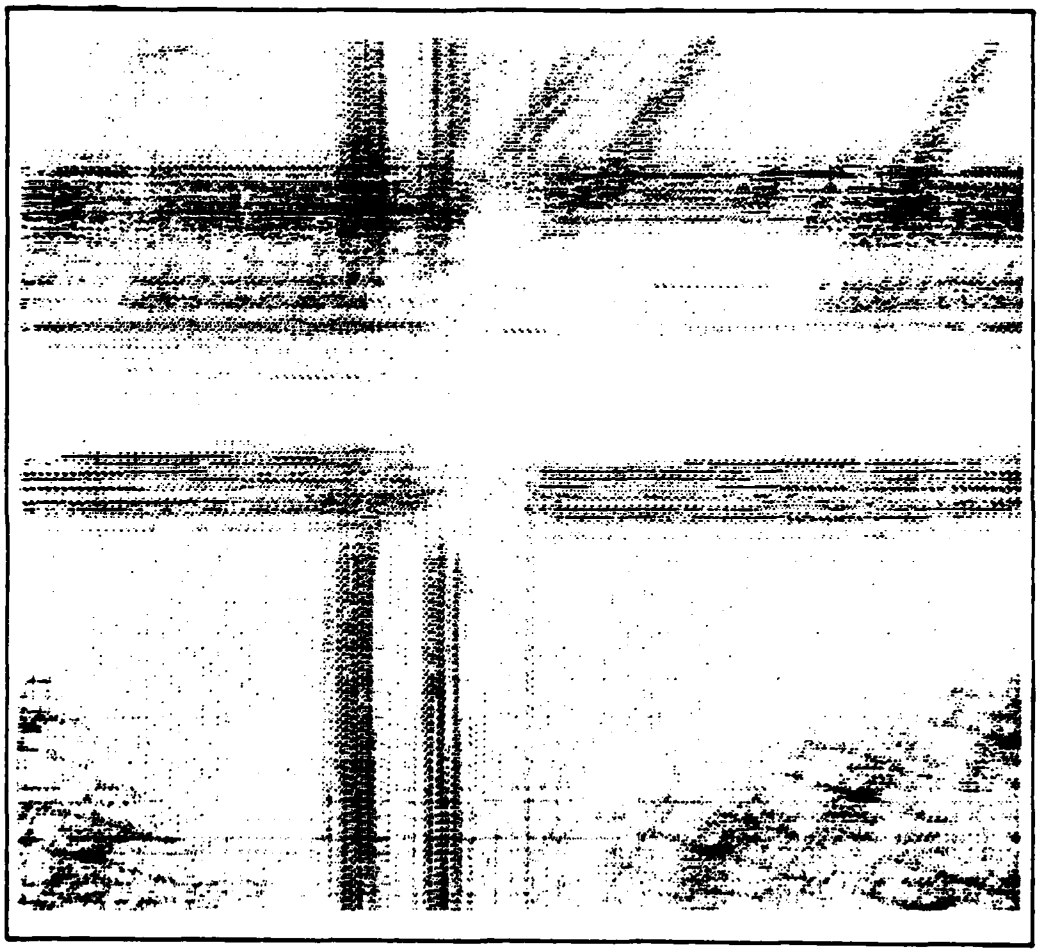
Combined Result of Wedge Operator on F16



F16 Template



Correlation of F16



Separated Orientation Correlation of F16

APPENDIX D

CLUSTER RECOGNITION PROGRAMS

```
C      PROGRAM HAM1 (SPATIAL CLUSTER RECOGNITION ALGORITHM)
C                          (PARTIALLY IMPLEMENTED HERE)

       INTEGER MAIN(7),IFLNM(7),F2(7),F3(7),MS(2),S1(2),S2(2),S3(2)
       REAL INMAT(128,32),OUTMAT(128,16),ENPCT,EDPCT,ENTHR,EDTHR

       CALL IOF(1,MAIN,IFLNM,F2,F3,MS,S1,S2,S3)

       CALL OPEN(1,IFLNM,1,IER)
       CALL CHECK(IER)
       CALL OPEN(2,"ENHC.IR",2,IER)
       CALL CHECK(IER)
       CALL OPEN(3,"NTHR.IR",3,IER)
       CALL CHECK(IER)
       CALL OPEN(4,"EDGE.IR",2,IER)
       CALL CHECK(IER)
       CALL OPEN(5,"DTHR.IR",3,IER)
       CALL CHECK(IER)
       CALL OPEN(6,"CTHR.IR",2,IER)
       CALL CHECK(IER)
       CALL OPEN(7,"CONN.IR",2,IER)
       CALL CHECK(IER)

C      ENHANCED IMAGE IS FORMED FROM ORIGINAL IMAGE

       CALL NHANCE(1,2,INMAT,OUTMAT)

       ENPCT=1.0

       CALL THRESH(2,3,ENPCT,ENTHR,INMAT)

C      EDGED IMAGE IS FORMED FROM ENHANCED IMAGE

       CALL EGIMAG(2,4,INMAT,OUTMAT)

       EDPCT=1.0

       CALL THRESH(4,5,EDPCT,EDTHR,INMAT)

C      EDGED IMAGE IS REDUCED BY CONJUNCTIVE THRESHOLDING

       CALL CONJUN(2,4,6,ENTHR,EDTHR,INMAT,OUTMAT)

C      EDGED IMAGE IS REDUCED BY CONNECTEDNESS TEST

       CALL CONNEC(6,7,INMAT,OUTMAT)

       CALL RESET
       STOP"<7><7><7><7>HAM1"
       END
```

```
      SUBROUTINE NHANCE(INFIL,OUTFIL,INMAT,OUTMAT)

      INTEGER INFIL,OUTFIL,TOGGLE
      REAL INMAT(128,32),OUTMAT(128,16)

      DO 1 K=1,8  ; FILL FIRST AND LAST 8 ROWS OF
         DO 1 J=1,128  ; RESULTS WITH ZERO
            OUTMAT(J,K)=0.0
1     CONTINUE
      CALL WRBLK(OUTFIL,0,OUTMAT,8,IER)
      CALL CHECK(IER)
      CALL WRBLK(OUTFIL,120,OUTMAT,8,IER)
      CALL CHECK(IER)

      CALL RDBLK(INFIL,0,INMAT,16,IER) ;READ IN FIRST 16 ROWS
      CALL CHECK(IER)

      TOGGLE=0  ; FLAG TO SHOW BUFFER WRAP-AROUND

      DO 5 I=1,7 ; COUNTER FOR 4K BUFFER LOADS

         CALL RDBLK(INFIL,16*I,INMAT(1,17-16*TOGGLE),16,IER)
         CALL CHECK(IER)                    ; READ IN NEXT 16 ROWS

         DO 3 K=1,16 ; COUNTER FOR OUTPUT BUFFER ROWS

            DO 2 J=1,8 ; ZERO FILL FIRST/LAST 8 COLUMNS
               OUTMAT(J,K)=0
               OUTMAT(J+120,K)=0
2           CONTINUE

            DO 3 J=9,120  ; ENHANCE THIS ROW
               CALL NHANC1(J,K,TOGGLE,INMAT,OUTMAT)
3        CONTINUE

         CALL WRBLK(OUTFIL,16*I-8,OUTMAT,16,IER) ; WRITE 16 ROWS
         CALL CHECK(IER)                         ; OF RESULTS

         IF(TOGGLE.EQ.1)GO TO 4
            TOGGLE=1     ; TOGGLE THE BUFFER WRAP-AROUND FLAG
            GO TO 5
4           TOGGLE=0

5     CONTINUE

      TYPE"ENHANCED IMAGE COMPLETE"

      RETURN
      END
```

```
      SUBROUTINE NHANC1(J,K,TOGGLE,INMAT,OUTMAT)

      INTEGER J,K,TOGGLE,J1,J2,J3,K1,K2,K3
      REAL INMAT(128,32),OUTMAT(128,16),A,B,C,D,E,F,G,H

      IF(TOGGLE.EQ.1)GO TO 1
         K1=K+7
         K2=K+8
         K3=K+9
         GO TO 2
1        K1=MOD(K+22,32)+1
         K2=MOD(K+23,32)+1
         K3=MOD(K+24,32)+1

2     J1=J-1
      J2=J
      J3=J+1

      A=10*(INMAT(J1,K1)+INMAT(J2,K1)+INMAT(J3,K1))-
     +      (INMAT(J3,K2)+INMAT(J3,K3)+INMAT(J2,K3)+
     +       INMAT(J1,K3)+INMAT(J1,K2))

      B=10*(INMAT(J2,K1)+INMAT(J3,K1)+INMAT(J3,K2))-
     +      (INMAT(J3,K3)+INMAT(J2,K3)+INMAT(J1,K3)+
     +       INMAT(J1,K2)+INMAT(J1,K1))

      C=10*(INMAT(J3,K1)+INMAT(J3,K2)+INMAT(J3,K3))-
     +      (INMAT(J2,K3)+INMAT(J1,K3)+INMAT(J1,K2)+
     +       INMAT(J1,K1)+INMAT(J2,K1))

      D=10*(INMAT(J3,K2)+INMAT(J3,K3)+INMAT(J2,K3))-
     +      (INMAT(J1,K3)+INMAT(J1,K2)+INMAT(J1,K1)+
     +       INMAT(J2,K1)+INMAT(J3,K1))

      E=10*(INMAT(J3,K3)+INMAT(J2,K3)+INMAT(J1,K3))-
     +      (INMAT(J1,K2)+INMAT(J1,K1)+INMAT(J2,K1)+
     +       INMAT(J3,K1)+INMAT(J3,K2))

      F=10*(INMAT(J2,K3)+INMAT(J1,K3)+INMAT(J1,K2))-
     +      (INMAT(J1,K1)+INMAT(J2,K1)+INMAT(J3,K1)+
     +       INMAT(J3,K2)+INMAT(J3,K3))

      G=10*(INMAT(J1,K3)+INMAT(J1,K2)+INMAT(J1,K1))-
     +      (INMAT(J2,K1)+INMAT(J3,K1)+INMAT(J3,K2)+
     +       INMAT(J3,K3)+INMAT(J2,K3))

      H=10*(INMAT(J1,K2)+INMAT(J1,K1)+INMAT(J2,K1))-
     +      (INMAT(J3,K1)+INMAT(J3,K2)+INMAT(J3,K3)+
     +       INMAT(J2,K3)+INMAT(J1,K3))

      OUTMAT(J,K)=AMAX1(A,B,C,D,E,F,G,H)

      RETURN
      END
```

```
      SUBROUTINE THRESH(INFIL,OUTFIL,PCT,THRLD,INMAT)

      INTEGER INFIL,OUTFIL
      REAL PCT,THRLD,INMAT(128,32),RSUM,AVG,DEV

      RSUM=0.0              ;  GET SET TO FIND AVERAGE
      CALL RDBLK(INFIL,16,INMAT,32,IER)
      CALL CHECK(IER)
      DO 1 K=3,32 ; COUNTER FOR INPUT BUFFER ROWS
         DO 1 J=17,112 ; COLUMN COUNT
            RSUM=RSUM+INMAT(J,K)
    1 CONTINUE
      CALL RDBLK(INFIL,48,INMAT,32,IER)
      CALL CHECK(IER)
      DO 2 K=1,32 ; COUNTER FOR INPUT BUFFER ROWS
         DO 2 J=17,112 ; COLUMN COUNT
            RSUM=RSUM+INMAT(J,K)
    2 CONTINUE
      CALL RDBLK(INFIL,80,INMAT,32,IER)
      CALL CHECK(IER)
      DO 3 K=1,30 ; COUNTER FOR INPUT BUFFER ROWS
         DO 3 J=17,112 ; COLUMN COUNT
            RSUM=RSUM+INMAT(J,K)
    3 CONTINUE

      AVG=RSUM/(96*92)

      RSUM=0.0              ;  GET SET TO FIND DEVIATION
      CALL RDBLK(INFIL,16,INMAT,32,IER)
      CALL CHECK(IER)
      DO 4 K=3,32  ; COUNTER FOR INPUT BUFFER ROWS
         DO 4 J=17,112 ; COLUMN COUNT
            RSUM=RSUM+(INMAT(J,K)-AVG)**2
    4 CONTINUE
      CALL RDBLK(INFIL,48,INMAT,32,IER)
      CALL CHECK(IER)
      DO 5 K=1,32  ; COUNTER FOR INPUT BUFFER ROWS
         DO 5 J=17,112 ; COLUMN COUNT
            RSUM=RSUM+(INMAT(J,K)-AVG)**2
    5 CONTINUE
      CALL RDBLK(INFIL,80,INMAT,32,IER)
      CALL CHECK(IER)
      DO 6 K=1,30  ; COUNTER FOR INPUT BUFFER ROWS
         DO 6 J=17,112 ; COLUMN COUNT
            RSUM=RSUM+(INMAT(J,K)-AVG)**2
    6 CONTINUE

      DEV=SQRT(RSUM/(96*92))

      THRLD=AVG+PCT*DEV

      TYPE"THRESHOLD = ",THRLD

      DO 8 I=0,3

         CALL RDBLK(INFIL,32*I,INMAT,32,IER)
         CALL CHECK(IER)

         DO 7 K=1,32
            DO 7 J=1,128
               IF(INMAT(J,K).LE.THRLD)INMAT(J,K)=0.0
    7    CONTINUE

         CALL WRBLK(OUTFIL,32*I,INMAT,32,IER)
         CALL CHECK(IER)

    8 CONTINUE

      RETURN
      END
```

```
      SUBROUTINE EGIMAG(INFIL,OUTFIL,INMAT,OUTMAT)

      INTEGER INFIL,OUTFIL,TOGGLE
      REAL INMAT(128,32),OUTMAT(128,16)

      DO 1 K=1,8  ; FILL FIRST AND LAST 8 ROWS OF
         DO 1 J=1,128  ; RESULTS WITH ZERO
            OUTMAT(J,K)=0.0
    1 CONTINUE
      CALL WRBLK(OUTFIL,0,OUTMAT,8,IER)
      CALL CHECK(IER)
      CALL WRBLK(OUTFIL,120,OUTMAT,8,IER)
      CALL CHECK(IER)

      CALL RDBLK(INFIL,0,INMAT,16,IER) ;READ IN FIRST 16 ROWS
      CALL CHECK(IER)

      TOGGLE=0  ; FLAG TO SHOW BUFFER WRAP-AROUND

      DO 5 I=1,7 ; COUNTER FOR 4K BUFFER LOADS

         CALL RDBLK(INFIL,16*I,INMAT(1,17-16*TOGGLE),16,IER)
         CALL CHECK(IER)                       ; READ IN NEXT 16 ROWS

         DO 3 K=1,16 ; COUNTER FOR OUTPUT BUFFER ROWS

            DO 2 J=1,8 ; ZERO FILL FIRST/LAST 8 COLUMNS
               OUTMAT(J,K)=0
               OUTMAT(J+120,K)=0
    2       CONTINUE

            DO 3 J=9,120  ; ENHANCE THIS ROW
               CALL EDGER1(J,K,TOGGLE,INMAT,OUTMAT)
    3    CONTINUE

         CALL WRBLK(OUTFIL,16*I-8,OUTMAT,16,IER) ; WRITE 16 ROWS
         CALL CHECK(IER)                         ; OF RESULTS

         IF(TOGGLE.EQ.1)GO TO 4
            TOGGLE=1     ; TOGGLE THE BUFFER WRAP-AROUND FLAG
            GO TO 5
    4       TOGGLE=0

    5 CONTINUE

      TYPE"EDGED IMAGE COMPLETE"

      RETURN
      END
```

```
      SUBROUTINE EDGER1(J,K,TOGGLE,INMAT,OUTMAT)

      INTEGER J,K,TOGGLE,J1,J2,J3,K1,K2,K3
      REAL INMAT(128,32),OUTMAT(128,16),A,B,C,D,E,F,G,H

      IF(TOGGLE.EQ.1)GO TO 1
         K1=K+7
         K2=K+8
         K3=K+9
         GO TO 2
1        K1=MOD(K+22,32)+1
         K2=MOD(K+23,32)+1
         K3=MOD(K+24,32)+1

2     J1=J-1
      J2=J
      J3=J+1

      A=5*(INMAT(J1,K1)+INMAT(J2,K1)+INMAT(J3,K1))-
     +  3*(INMAT(J3,K2)+INMAT(J3,K3)+INMAT(J2,K3)+
     +      INMAT(J1,K3)+INMAT(J1,K2))

      B=5*(INMAT(J2,K1)+INMAT(J3,K1)+INMAT(J3,K2))-
     +  3*(INMAT(J3,K3)+INMAT(J2,K3)+INMAT(J1,K3)+
     +      INMAT(J1,K2)+INMAT(J1,K1))

      C=5*(INMAT(J3,K1)+INMAT(J3,K2)+INMAT(J3,K3))-
     +  3*(INMAT(J2,K3)+INMAT(J1,K3)+INMAT(J1,K2)+
     +      INMAT(J1,K1)+INMAT(J2,K1))

      D=5*(INMAT(J3,K2)+INMAT(J3,K3)+INMAT(J2,K3))-
     +  3*(INMAT(J1,K3)+INMAT(J1,K2)+INMAT(J1,K1)+
     +      INMAT(J2,K1)+INMAT(J3,K1))

      E=5*(INMAT(J3,K3)+INMAT(J2,K3)+INMAT(J1,K3))-
     +  3*(INMAT(J1,K2)+INMAT(J1,K1)+INMAT(J2,K1)+
     +      INMAT(J3,K1)+INMAT(J3,K2))

      F=5*(INMAT(J2,K3)+INMAT(J1,K3)+INMAT(J1,K2))-
     +  3*(INMAT(J1,K1)+INMAT(J2,K1)+INMAT(J3,K1)+
     +      INMAT(J3,K2)+INMAT(J3,K3))

      G=5*(INMAT(J1,K3)+INMAT(J1,K2)+INMAT(J1,K1))-
     +  3*(INMAT(J2,K1)+INMAT(J3,K1)+INMAT(J3,K2)+
     +      INMAT(J3,K3)+INMAT(J2,K3))

      H=5*(INMAT(J1,K2)+INMAT(J1,K1)+INMAT(J2,K1))-
     +  3*(INMAT(J3,K1)+INMAT(J3,K2)+INMAT(J3,K3)+
     +      INMAT(J2,K3)+INMAT(J1,K3))

      OUTMAT(J,K)=AMAX1(A,B,C,D,E,F,G,H)

      RETURN
      END
```

```
      SUBROUTINE CONJUN(FILE1,FILE2,FILE3,ENTHR,EDTHR,INMAT,OUTMAT)

      INTEGER FILE1,FILE2,FILE3
      REAL ENTHR,EDTHR,INMAT(128,32),OUTMAT(128,16)

      DO 1 K=1,16                  ; ZERO FIRST/LAST 16 ROWS
         DO 1 J=1,128
            OUTMAT(J,K)=0.0
    1 CONTINUE
      CALL WRBLK(FILE3,0,OUTMAT,16,IER)
      CALL CHECK(IER)
      CALL WRBLK(FILE3,112,OUTMAT,16,IER)
      CALL CHECK(IER)

      DO 5 I=1,6 ; COUNTER FOR 4K BUFFER LOADS

         CALL RDBLK(FILE1,16*I,INMAT(1,1),16,IER) ; READ 16 ROWS
         CALL CHECK(IER)
         CALL RDBLK(FILE2,16*I,INMAT(1,17),16,IER) ; READ 16 ROWS
         CALL CHECK(IER)

         DO 4 K=1,16 ; COUNTER FOR OUTPUT BUFFER ROWS

            DO 3 J=17,112 ; APPLY CONJUNCTIVE THRESHOLD TEST
               IF(INMAT(J,K).GT.ENTHR.AND.
     +            INMAT(J,K+16).GT.EDTHR)GO TO 2
                  OUTMAT(J,K)=0.0
                  GO TO 3
    2             OUTMAT(J,K)=15.0
    3       CONTINUE

            DO 4 J=1,16    ; ZERO THE PERIMETER
               OUTMAT(J,K)=0
               OUTMAT(J+112,K)=0

    4    CONTINUE

         CALL WRBLK(FILE3,16*I,OUTMAT,16,IER) ; WRITE 16 ROWS
         CALL CHECK(IER)

    5 CONTINUE

      DO 6 K=1,2              ; ZERO THE PERIMETER
         DO 6 J=1,128
            OUTMAT(J,1)=0.0
    6 CONTINUE
      CALL WRBLK(FILE3,16,OUTMAT,2,IER)
      CALL CHECK(IER)
      CALL WRBLK(FILE3,110,OUTMAT,2,IER)
      CALL CHECK(IER)

      TYPE"CONJUNCTIVE THRESHOLDING DONE"

      RETURN
      END
```

```
      SUBROUTINE CONNEC(INFIL,OUTFIL,INMAT,OUTMAT)

      INTEGER INFIL,OUTFIL,TOGGLE,ICNT,JCNT
      REAL INMAT(128,32),OUTMAT(128,16)

      JCNT=1   ; COUNTER FOR TEST LOOP COUNTING
    1 ICNT=0   ; COUNTER FOR TEST STABILITY

      CALL RDBLK(INFIL,0,INMAT,16,IER) ;READ IN FIRST 16 ROWS
      CALL CHECK(IER)

      IF(JCNT.GT.1)INFIL=OUTFIL ;OPERATE ON SAME FILE AFTER FIRST LOOP
      TOGGLE=0   ; FLAG TO SHOW BUFFER WRAP-AROUND

      DO 5 I=1,7 ; COUNTER FOR 4K BUFFER LOADS

         CALL RDBLK(INFIL,16*I,INMAT(1,17-16*TOGGLE),16,IER)
         CALL CHECK(IER)                ; READ NEXT 16 ROWS

         DO 3 K=1,16 ; COUNTER FOR OUTPUT BUFFER ROWS

            DO 2 J=1,8 ; ZERO FILL FIRST/LAST 8 COLUMNS
               OUTMAT(J,K)=0.0
               OUTMAT(J+120,K)=0.0
    2       CONTINUE

            DO 3 J=9,120 ; TEST THIS ROW FOR CONNECTIVITY
               CALL CONEC1(J,K,TOGGLE,ICNT,INMAT,OUTMAT)

    3    CONTINUE

         CALL WRBLK(OUTFIL,16*I-8,OUTMAT,16,IER) ; WRITE 16 ROWS
         CALL CHECK(IER)                          ; OF RESULTS

         IF(TOGGLE.EQ.1)GO TO 4
            TOGGLE=1     ; TOGGLE THE BUFFER WRAP-AROUND FLAG
            GO TO 5
    4       TOGGLE=0

    5 CONTINUE

      DO 6 K=1,8    ; ZERO FILL FIRST/LAST 8 ROWS
         DO 6 J=1,128
            OUTMAT(J,K)=0.0
    6 CONTINUE
      CALL WRBLK(OUTFIL,0,OUTMAT,8,IER)
      CALL CHECK(IER)
      CALL WRBLK(OUTFIL,120,OUTMAT,8,IER)
      CALL CHECK(IER)

      JCNT=JCNT+1

      IF(ICNT.NE.0)GO TO 1 ; CONNECTIVITY TEST HAS NOT STABILIZED

      TYPE"CONNECTIVITY TEST DONE AT LOOP #",JCNT-1

      RETURN
      END
```

```
      SUBROUTINE CONEC1(J,K,TOGGLE,ICNT,INMAT,OUTMAT)

      INTEGER J,K,TOGGLE,ICNT
      REAL INMAT(128,32),OUTMAT(128,16)
      INTEGER J1,J2,J3,K1,K2,K3,NCNT

      IF(TOGGLE.EQ.1)GO TO 1
         K1=K+7
         K2=K+8
         K3=K+9
         GO TO 2

 1       K1=MOD(K+22,32)+1
         K2=MOD(K+23,32)+1
         K3=MOD(K+24,32)+1

 2    J1=J-1
      J2=J
      J3=J+1

      IF(INMAT(J2,K2).EQ.0.0)GO TO 4

      NCNT=0

      IF(INMAT(J2,K1).NE.0.0)NCNT=NCNT+1
      IF(INMAT(J3,K2).NE.0.0)NCNT=NCNT+1
      IF(INMAT(J2,K3).NE.0.0)NCNT=NCNT+1
      IF(INMAT(J1,K2).NE.0.0)NCNT=NCNT+1

      IF(NCNT.LT.2)GO TO 3
         OUTMAT(J,K)=15.0
         GO TO 5

 3       OUTMAT(J,K)=0
         ICNT=ICNT+1   ; INCREMENT COUNTER TO SHOW CHANGE
         GO TO 5

 4    OUTMAT(J,K)=0.0

 5    CONTINUE

      RETURN
      END
```

```
C     PROGRAM HAM21 (SPATIAL CLUSTER RECOGNITION ALGORITHM)

      INTEGER MAIN(7),IFLNM(7),F2(7),F3(7),MS(2),S1(2),S2(2),S3(2)
      REAL INMAT(128,32),OUTMAT(128,16),ENPCT,EDPCT

      CALL IOF(1,MAIN,IFLNM,F2,F3,MS,S1,S2,S3)

      CALL OPEN(1,IFLNM,1,IER)
      CALL CHECK(IER)
      CALL OPEN(2,"ENHC.IR",2,IER)
      CALL CHECK(IER)
      CALL OPEN(3,"NTHR.TH",2,IER)
      CALL CHECK(IER)
      CALL OPEN(4,"NTHR.IR",3,IER)
      CALL CHECK(IER)
      CALL OPEN(5,"EDGE.IR",2,IER)
      CALL CHECK(IER)
      CALL OPEN(6,"DTHR.TH",2,IER)
      CALL CHECK(IER)
      CALL OPEN(7,"DTHR.IR",3,IER)
      CALL CHECK(IER)
      CALL OPEN(8,"CTHR.IR",2,IER)
      CALL CHECK(IER)
      CALL OPEN(9,"CONN.IR",2,IER)
      CALL CHECK(IER)

C     ENHANCED IMAGE IS FORMED FROM ORIGINAL IMAGE

      CALL NHANCE(1,2,INMAT,OUTMAT)

      ENPCT=1.0

      MASKSZ=17

      CALL LTHRSH(2,3,4,MASKSZ,ENPCT,INMAT,OUTMAT)

C     EDGED IMAGE IS FORMED FROM ORIGINAL IMAGE

      CALL EGIMAG(1,5,INMAT,OUTMAT)

      EDPCT=1.0

      CALL LTHRSH(5,6,7,MASKSZ,EDPCT,INMAT,OUTMAT)

C     EDGED IMAGE IS REDUCED BY CONJUNCTIVE THRESHOLDING

      CALL LOCONJ(2,3,5,6,8,INMAT,OUTMAT)

C     EDGED IMAGE IS REDUCED BY CONNECTEDNESS TEST

      CALL CONNEC(8,9,INMAT,OUTMAT)

      CALL RESET
      STOP"<7><7><7><7>HAM21"
      END
```

```
      SUBROUTINE LTHRSH(INFILE,OFILE1,OFILE2,MASKSZ,PCT,INMAT,OUTMAT)

      INTEGER INFILE,OFILE1,OFILE2,MASKSZ,TOGGLE,OFF
      REAL PCT,INMAT(128,32),OUTMAT(128,16),MSUM,NRSUM,CHANGE

      DO 1 K=1,8                ; FILL FIRST AND LAST 8 ROWS OF
         DO 1 J=1,128           ; THRESHOLD IMAGE WITH CONSTANT
            OUTMAT(J,K)=1.0
    1 CONTINUE
      CALL WRBLK(OFILE1,0,OUTMAT,8,IER)
      CALL CHECK(IER)
      CALL WRBLK(OFILE1,120,OUTMAT,8,IER)
      CALL CHECK(IER)

      CALL RDBLK(INFILE,0,INMAT,32,IER) ;READ IN FIRST 32 ROWS
      CALL CHECK(IER)

      OFF=(MASKSZ-1)/2
      MSUM=0.0
      DO 2 K=0,MASKSZ-1   ; INITIALIZE AVERAGING VARIABLES
         DO 2 J=0,MASKSZ-1
            MSUM=MSUM+INMAT(9-OFF+J,9-OFF+K)
    2 CONTINUE
      CHANGE=0
      DO 3 J=0,MASKSZ-1
         CHANGE=CHANGE+INMAT(9-OFF+J,9+OFF)
    3 CONTINUE
      NRSUM=MSUM-CHANGE
      TOGGLE=0   ; FLAG TO SHOW BUFFER WRAP-AROUND

      DO 4 K=1,16   ; COMPUTE THRESHOLD FOR FIRST 16 ROWS (4K)
         CALL THROW1(K,MASKSZ,TOGGLE,NRSUM,PCT,INMAT,OUTMAT)
    4 CONTINUE
      CALL WRBLK(OFILE1,8,OUTMAT,16,IER) ; WRITE RESULTS
      CALL CHECK(IER)

      TOGGLE=1
      DO 7 I=2,7   ; COUNTER FOR 4K BUFFER LOADS
         CALL RDBLK(INFILE,16*I,INMAT(1,17-16*TOGGLE),16,IER)
         CALL CHECK(IER)   ; READ IN NEXT 16 ROWS

         DO 5 K=1,16 ; COUNTER FOR OUTPUT BUFFER ROWS
            CALL THROW1(K,MASKSZ,TOGGLE,NRSUM,PCT,INMAT,OUTMAT)
    5    CONTINUE
         CALL WRBLK(OFILE1,16*I-8,OUTMAT,16,IER) ; WRITE 16 ROWS
         CALL CHECK(IER)

         IF(TOGGLE.EQ.1)GO TO 6
            TOGGLE=1      ; TOGGLE THE BUFFER WRAP-AROUND FLAG
            GO TO 7
    6       TOGGLE=0
    7 CONTINUE

      DO 9 I=0,7          ; FORM THRESHOLDED IMAGE

         CALL RDBLK(OFILE1,16*I,INMAT,16,IER)
         CALL CHECK(IER)
         CALL RDBLK(INFILE,16*I,OUTMAT,16,IER)
         CALL CHECK(IER)
```

```
      DO 8 K=1,16
         DO 8 J=1,128
            IF(OUTMAT(J,K).LE.INMAT(J,K))OUTMAT(J,K)=0.0
8     CONTINUE

      CALL WRBLK(OFILE2,16*I,OUTMAT,16,IER)
      CALL CHECK(IER)

9  CONTINUE

   TYPE"THRESHOLD MATRIX COMPLETE"
   RETURN
   END
```

```
      SUBROUTINE THROW1(K, MASKSZ, TOGGLE, NRSUM, PCT, INMAT, OUTMAT)

      INTEGER K, MASKSZ, TOGGLE, OFF
      REAL NRSUM, PCT, INMAT(128, 32), OUTMAT(128, 16)
      REAL FCOLSM, CHANGE, ASUM, AVG, DSUM, DEV

      IF(TOGGLE.EQ.1)GO TO 1   ; INITIALIZE INDEXING VARIABLES
         K1=7
         GO TO 2
1        K1=23
2     CONTINUE
      OFF=(MASKSZ-1)/2

      DO 3 J=1,8    ; FILL FIRST/LAST EIGHT PIXELS WITH CONSTANT
         OUTMAT(J,K)=1.0
         OUTMAT(J+120,K)=1.0
3     CONTINUE

      CHANGE=0.0          ; FIND NEW MATRIX SUM
      DO 4 I=0,MASKSZ-1
         CHANGE=CHANGE+INMAT(9-OFF+I,MOD(K+OFF+K1,32)+1)
4     CONTINUE
      ASUM=NRSUM+CHANGE

      CHANGE=0.0          ; FIND NEW NEXT ROW SUM
      DO 5 I=0,MASKSZ-1
         CHANGE=CHANGE+INMAT(9-OFF+I,MOD(K-OFF+K1,32)+1)
5     CONTINUE
      NRSUM=ASUM-CHANGE

      FCOLSM=0.0          ; INITIALIZE FCOLSM
      DO 6 I=0,MASKSZ-1
         FCOLSM=FCOLSM+INMAT(9+OFF,MOD(K-OFF+I+K1,32)+1)
6     CONTINUE

      DO 10 J=9,120   ; COLUMN COUNT

         CHANGE=0.0        ; FIND NEW MATRIX SUM AND AVERAGE
         DO 7 I=0,MASKSZ-1
            CHANGE=CHANGE+INMAT(J+OFF,MOD(K-OFF+I+K1,32)+1)
7        CONTINUE
         ASUM=ASUM+CHANGE-FCOLSM
         AVG=ASUM/MASKSZ**2

         FCOLSM=0.0        ; FIND NEW FIRST COLUMN SUM
         DO 8 I=0,MASKSZ-1
            FCOLSM=FCOLSM+INMAT(J-OFF,MOD(K-OFF+I+K1,32)+1)
8        CONTINUE

         DSUM=0.0    ; FIND NEW DEVIATION
         DO 9 I1=0,MASKSZ-1
            DO 9 I2=0,MASKSZ-1
               DSUM=DSUM+(INMAT(J-OFF+I1,MOD(K-OFF+I2+K1,32)+1)-AVG)**2
9        CONTINUE
         DEV=SQRT(DSUM/MASKSZ**2)

         OUTMAT(J,K)=AVG+PCT*DEV

10    CONTINUE

      RETURN
      END
```

```
      SUBROUTINE LOCONJ(FILE1,FILE2,FILE3,FILE4,FILE5,INMAT,OUTMAT)

      INTEGER FILE1,FILE2,FILE3,FILE4,FILE5
      REAL INMAT(128,32),OUTMAT(128,16)

      DO 1 K=1,16     ; ZERO FIRST/LAST 16 ROWS
         DO 1 J=1,128
            OUTMAT(J,K)=0.0
1     CONTINUE
      CALL WRBLK(FILE5,0,OUTMAT,16,IER)
      CALL CHECK(IER)
      CALL WRBLK(FILE5,112,OUTMAT,16,IER)
      CALL CHECK(IER)

      DO 5 I1=1,6   ; COUNTER FOR 4K BUFFER OUTPUT

         DO 4 I2=0,1 ; COUNTER FOR 2K BUFFER INPUT

            CALL RDBLK(FILE1,16*I1+8*I2,INMAT(1,1),8,IER) ; READ 8 ROWS
            CALL CHECK(IER)
            CALL RDBLK(FILE2,16*I1+8*I2,INMAT(1,9),8,IER) ; READ 8 ROWS
            CALL CHECK(IER)
            CALL RDBLK(FILE3,16*I1+8*I2,INMAT(1,17),8,IER) ; READ 8 ROWS
            CALL CHECK(IER)
            CALL RDBLK(FILE4,16*I1+8*I2,INMAT(1,25),8,IER) ; READ 8 ROWS
            CALL CHECK(IER)

            DO 4 K=1,8 ; COUNTER FOR OUTPUT BUFFER ROWS

               DO 3 J=17,112 ; APPLY CONJUNCTIVE THRESHOLD TEST

                  IF(INMAT(J,K).GT.INMAT(J,K+8).AND.
     +               INMAT(J,K+16).GT.INMAT(J,K+24))GO TO 2
                     OUTMAT(J,K+8*I2)=0.0
                     GO TO 3
2                    OUTMAT(J,K+8*I2)=15.0
3              CONTINUE

               DO 4 J=1,16   ; ZERO THE PERIMETER
                  OUTMAT(J,K+8*I2)=0.0
                  OUTMAT(J+112,K+8*I2)=0.0

4        CONTINUE

         CALL WRBLK(FILE5,16*I1,OUTMAT,16,IER) ; WRITE 16 ROWS
         CALL CHECK(IER)

5     CONTINUE

      DO 6 K=1,2          ; ZERO THE PERIMETER
         DO 6 J=1,128
            OUTMAT(J,K)=0.0
6     CONTINUE
      CALL WRBLK(FILE5,16,OUTMAT,2,IER)
      CALL CHECK(IER)
      CALL WRBLK(FILE5,110,OUTMAT,2,IER)
      CALL CHECK(IER)

      TYPE"LOCAL CONJUNCTIVE THRESHOLDING DONE"

      RETURN
      END
```

```
C     PROGRAM HAM31 (SPATIAL CLUSTER RECOGNITION ALGORITHM)

      INTEGER MAIN(7),IFLNM(7),F2(7),F3(7),MS(2),S1(2),S2(2),S3(2)
      REAL INMAT(128,32),OUTMAT(128,16),PCT,DMIN,DMAX

      CALL IOF(1,MAIN,IFLNM,F2,F3,MS,S1,S2,S3)

      CALL OPEN(1,IFLNM,1,IER)
      CALL CHECK(IER)
      CALL OPEN(2,"ENHC.IR",2,IER)
      CALL CHECK(IER)
      CALL OPEN(3,"NTHR.TH",2,IER)
      CALL CHECK(IER)
      CALL OPEN(4,"NTHR.IR",3,IER)
      CALL CHECK(IER)
      CALL OPEN(5,"EDGE.IR",2,IER)
      CALL CHECK(IER)
      CALL OPEN(6,"DTHR.TH",2,IER)
      CALL CHECK(IER)
      CALL OPEN(7,"DTHR.IR",3,IER)
      CALL CHECK(IER)
      CALL OPEN(8,"CTHR.IR",2,IER)
      CALL CHECK(IER)
      CALL OPEN(9,"CONN.IR",2,IER)
      CALL CHECK(IER)

C     ENHANCED IMAGE IS FORMED FROM ORIGINAL IMAGE

      CALL NHANCE(1,2,INMAT,OUTMAT)

      MASKSZ=7
      PCT=1.0
      DMIN=100.0
      DMAX=150.0

      CALL LOTHRS(2,3,4,MASKSZ,PCT,DMIN,DMAX,INMAT,OUTMAT)

C     EDGED IMAGE IS FORMED FROM ORIGINAL IMAGE

      CALL EGIMAG(1,5,INMAT,OUTMAT)

      DMIN=0.0
      DMAX=10.0

      CALL LOTHRS(5,6,7,MASKSZ,PCT,DMIN,DMAX,INMAT,OUTMAT)

C     EDGED IMAGE IS REDUCED BY CONJUNCTIVE THRESHOLDING

      CALL LOCONJ(2,3,5,6,8,INMAT,OUTMAT)

C     EDGED IMAGE IS REDUCED BY CONNECTEDNESS TEST

      CALL CONNEC(8,9,INMAT,OUTMAT)

      CALL RESET
      STOP"<7><7><7><7>HAM31"
      END
```

```
      SUBROUTINE LOTHRS(INFILE,OFILE1,OFILE2,MASKSZ,PCT,DMIN,DMAX,
     +                  INMAT,OUTMAT)

      INTEGER INFILE,OFILE1,OFILE2,MASKSZ,TOGGLE,OFF
      REAL PCT,DMIN,DMAX,INMAT(128,32),OUTMAT(128,16),MSUM,NRSUM,CHANGE

      DO 1 K=1,8                ; FILL FIRST AND LAST 8 ROWS OF
         DO 1 J=1,128           ; THRESHOLD IMAGE WITH CONSTANT
            OUTMAT(J,K)=1.0
    1 CONTINUE
      CALL WRBLK(OFILE1,0,OUTMAT,8,IER)
      CALL CHECK(IER)
      CALL WRBLK(OFILE1,120,OUTMAT,8,IER)
      CALL CHECK(IER)

      CALL RDBLK(INFILE,0,INMAT,32,IER) ;READ IN FIRST 32 ROWS
      CALL CHECK(IER)

      OFF=(MASKSZ-1)/2
      MSUM=0.0
      DO 2 K=0,MASKSZ-1   ; INITIALIZE AVERAGING VARIABLES
         DO 2 J=0,MASKSZ-1
            MSUM=MSUM+INMAT(9-OFF+J,9-OFF+K)
    2 CONTINUE
      CHANGE=0
      DO 3 J=0,MASKSZ-1
         CHANGE=CHANGE+INMAT(9-OFF+J,9+OFF)
    3 CONTINUE
      NRSUM=MSUM-CHANGE
      TOGGLE=0   ; FLAG TO SHOW BUFFER WRAP-AROUND

      DO 4 K=1,16   ; COMPUTE THRESHOLD FOR FIRST 16 ROWS (4K)
         CALL THROW2(K,MASKSZ,TOGGLE,NRSUM,PCT,DMIN,DMAX,INMAT,OUTMAT)
    4 CONTINUE
      CALL WRBLK(OFILE1,8,OUTMAT,16,IER) ; WRITE RESULTS
      CALL CHECK(IER)

      TOGGLE=1
      DO 7 I=2,7   ; COUNTER FOR 4K BUFFER LOADS
         CALL RDBLK(INFILE,16*I,INMAT(1,17-16*TOGGLE),16,IER)
         CALL CHECK(IER)   ; READ IN NEXT 16 ROWS

         DO 5 K=1,16 ; COUNTER FOR OUTPUT BUFFER ROWS
            CALL THROW2(K,MASKSZ,TOGGLE,NRSUM,PCT,DMIN,DMAX,
     +                  INMAT,OUTMAT)
    5    CONTINUE
         CALL WRBLK(OFILE1,16*I-8,OUTMAT,16,IER) ; WRITE 16 ROWS
         CALL CHECK(IER)

         IF(TOGGLE.EQ.1)GO TO 6
            TOGGLE=1     ; TOGGLE THE BUFFER WRAP-AROUND FLAG
            GO TO 7
    6       TOGGLE=0
    7 CONTINUE

      DO 9 I=0,7          ; FORM THRESHOLDED IMAGE

         CALL RDBLK(OFILE1,16*I,INMAT,16,IER)
         CALL CHECK(IER)
         CALL RDBLK(INFILE,16*I,OUTMAT,16,IER)
```

```
      CALL CHECK(IER)

      DO 8 K=1,16
         DO 8 J=1,128
            IF(OUTMAT(J,K).LE.INMAT(J,K))OUTMAT(J,K)=0.0
8     CONTINUE

      CALL WRBLK(OFILE2,16*I,OUTMAT,16,IER)
      CALL CHECK(IER)

9   CONTINUE

    TYPE"THRESHOLD MATRIX COMPLETE"
    RETURN
    END
```

```
      SUBROUTINE THROW2(K,MASKSZ,TOGGLE,NRSUM,PCT,DMIN,DMAX,
     +                  INMAT,OUTMAT)

      INTEGER K,MASKSZ,TOGGLE,OFF
      REAL NRSUM,PCT,DMIN,DMAX,INMAT(128,32),OUTMAT(128,16)
      REAL FCOLSM,CHANGE,ASUM,AVG,DSUM,DEV

      IF(TOGGLE.EQ.1)GO TO 1  ; INITIALIZE INDEXING VARIABLES
         K1=7
         GO TO 2
1        K1=23
2     CONTINUE
      OFF=(MASKSZ-1)/2

      DO 3 J=1,8    ; FILL FIRST/LAST EIGHT PIXELS WITH CONSTANT
         OUTMAT(J,K)=1.0
         OUTMAT(J+120,K)=1.0
3     CONTINUE

      CHANGE=0.0         ; FIND NEW MATRIX SUM
      DO 4 I=0,MASKSZ-1
         CHANGE=CHANGE+INMAT(9-OFF+I,MOD(K+OFF+K1,32)+1)
4     CONTINUE
      ASUM=NRSUM+CHANGE

      CHANGE=0.0        ; FIND NEW NEXT ROW SUM
      DO 5 I=0,MASKSZ-1
         CHANGE=CHANGE+INMAT(9-OFF+I,MOD(K-OFF+K1,32)+1)
5     CONTINUE
      NRSUM=ASUM-CHANGE

      FCOLSM=0.0         ; INITIALIZE FCOLSM
      DO 6 I=0,MASKSZ-1
         FCOLSM=FCOLSM+INMAT(9+OFF,MOD(K-OFF+I+K1,32)+1)
6     CONTINUE

      DO 10 J=9,120   ; COLUMN COUNT

         CHANGE=0.0      ; FIND NEW MATRIX SUM AND AVERAGE
         DO 7 I=0,MASKSZ-1
            CHANGE=CHANGE+INMAT(J+OFF,MOD(K-OFF+I+K1,32)+1)
7        CONTINUE
         ASUM=ASUM+CHANGE-FCOLSM
         AVG=ASUM/MASKSZ**2

         FCOLSM=0.0      ; FIND NEW FIRST COLUMN SUM
         DO 8 I=0,MASKSZ-1
            FCOLSM=FCOLSM+INMAT(J-OFF,MOD(K-OFF+I+K1,32)+1)
8        CONTINUE

         DSUM=0.0    ; FIND NEW DEVIATION
         DO 9 I1=0,MASKSZ-1
            DO 9 I2=0,MASKSZ-1
               DSUM=DSUM+(INMAT(J-OFF+I1,MOD(K-OFF+I2+K1,32)+1)-AVG)**2
9        CONTINUE
         DEV=SQRT(DSUM/MASKSZ**2)

         DEV=DEV*PCT              ; MODIFY DEVIATION
         IF(DEV.LT.DMIN)DEV=DMIN
         IF(DEV.GT.DMAX)DEV=DMAX
         OUTMAT(J,K)=AVG+DEV

10    CONTINUE

      RETURN
      END
```

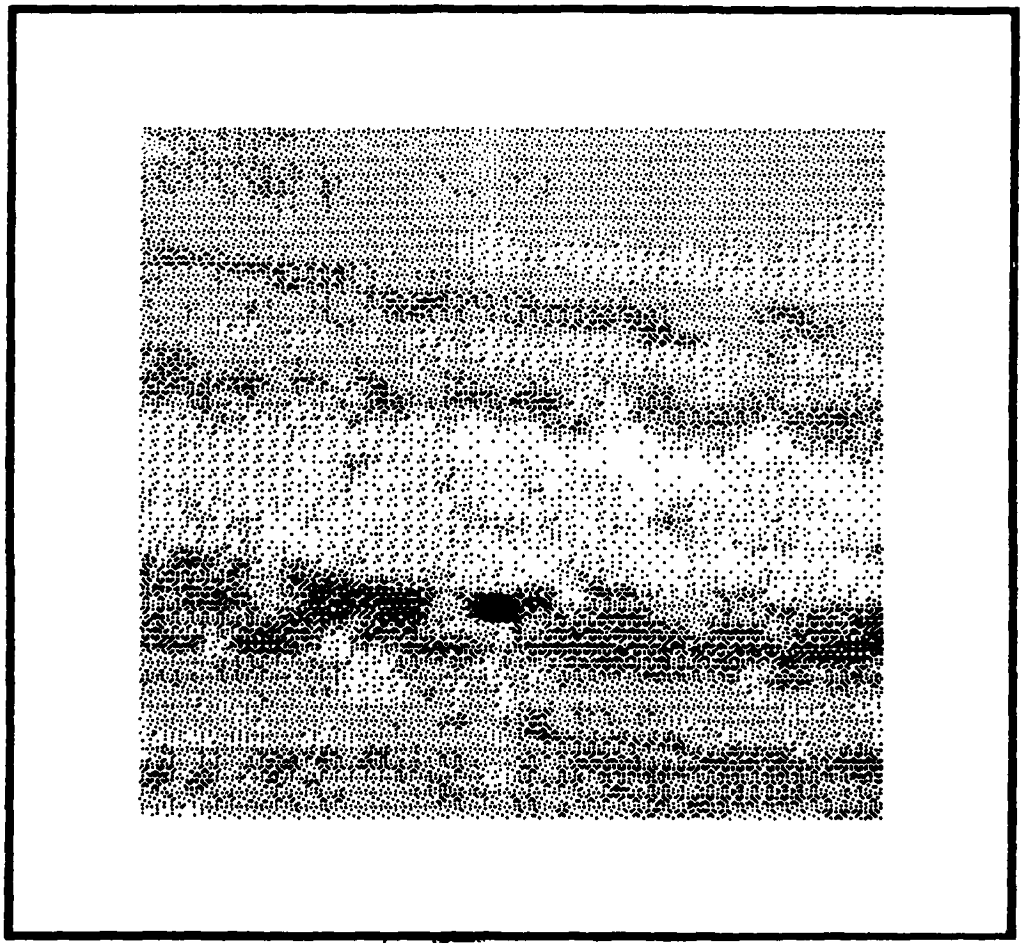
```
C     PROGRAM HAM61 (SPATIAL CLUSTER RECOGNITION ALGORITHM)

      INTEGER MAIN(7),IFLNM(7),F2(7),F3(7),MS(2),S1(2),S2(2),S3(2)
      REAL INMAT(128,32),ENHMAT(128,16),PCT,DMIN,DMAX

      CALL IOF(1,MAIN,IFLNM,F2,F3,MS,S1,S2,S3)

      CALL OPEN(1,IFLNM,1,IER)
      CALL CHECK(IER)
      CALL OPEN(2,"ENHC.IR",2,IER)
      CALL CHECK(IER)
      CALL OPEN(3,"NTHR.IR",2,IER)
      CALL CHECK(IER)
      CALL OPEN(4,"CONN.IR",2,IER)
      CALL CHECK(IER)

C     ENHANCED IMAGE FORMED FROM ORIGINAL IMAGE

      CALL NHANCE(1,2,INMAT,ENHMAT)

C     LOCAL THRESHOLDS COMPUTED FOR ENHANCED IMAGE
C         AND ENHANCED IMAGE LOCAL THRESHOLDED

      MASKSZ=7
      PCT=0.75
      DMIN=100.0
      DMAX=1000000.0

      CALL NTHRSH(2,3,MASKSZ,PCT,DMIN,DMAX,INMAT,ENHMAT)

C     THRESHOLDED IMAGE IS REDUCED BY CONNECTEDNESS TEST

      CALL CONNEC(3,4,INMAT,ENHMAT)

      CALL RESET
      STOP"<7><7><7><7>HAM61"
      END
```

```
      SUBROUTINE NTHRSH(ENHFIL, NIRFIL, MASKSZ, PCT, DMIN, DMAX, INMAT, ENHMAT)

      INTEGER ENHFIL, NIRFIL, MASKSZ
      REAL PCT, DMIN, DMAX, INMAT(128, 32), ENHMAT(128, 16)
      INTEGER OFF, TOGGLE
      REAL MSUM, NRMSUM, CHANGE

      CALL RDBLK(ENHFIL, 8, INMAT, 32, IER) ;READ IN FIRST 32 ROWS
      CALL CHECK(IER)                       ; OF ENHANCED IMAGE

      OFF=(MASKSZ-1)/2
      MSUM=0.0
      DO 1 K=0, MASKSZ-1   ; INITIALIZE AVERAGING VARIABLES
         DO 1 J=0, MASKSZ-1
            MSUM=MSUM+INMAT(17-OFF+J, 9-OFF+K)
    1 CONTINUE
      CHANGE=0.0
      DO 2 J=0, MASKSZ-1
         CHANGE=CHANGE+INMAT(17-OFF+J, 9+OFF)
    2 CONTINUE
      NRMSUM=MSUM-CHANGE

      TOGGLE=0  ; FLAG TO SHOW BUFFER WRAP-AROUND

      DO 3 K=1, 16    ; THRESHOLD FIRST 16 ROWS
         CALL NTROW1(K, MASKSZ, TOGGLE, NRMSUM, PCT, DMIN, DMAX, INMAT, ENHMAT)
    3 CONTINUE

      CALL WRBLK(NIRFIL, 16, ENHMAT, 16, IER) ;  WRITE RESULTS
      CALL CHECK(IER)

      TOGGLE=1
      DO 6 I=2, 6   ; THRESHOLD ENHANCED IMAGE

         CALL RDBLK(ENHFIL, 16*I+8, INMAT(1, 17-16*TOGGLE), 16, IER)
         CALL CHECK(IER)   ; READ IN NEXT 16 ROWS

         DO 4 K=1, 16     ; COUNTER FOR OUTPUT BUFFER ROWS
            CALL NTROW1(K, MASKSZ, TOGGLE, NRMSUM, PCT, DMIN, DMAX,
     +                  INMAT, ENHMAT)
    4    CONTINUE

         CALL WRBLK(NIRFIL, 16*I, ENHMAT, 16, IER) ;  WRITE 16 ROWS
         CALL CHECK(IER)

         IF(TOGGLE.EQ.1)GO TO 5
            TOGGLE=1      ; TOGGLE THE BUFFER WRAP-AROUND FLAG
            GO TO 6
    5       TOGGLE=0

    6 CONTINUE

      DO 7 K=1, 16      ; ZERO FILL FIRST/LAST 18 ROWS
         DO 7 J=1, 128
            ENHMAT(J, K)=0.0
    7 CONTINUE
      CALL WRBLK(NIRFIL, 0, ENHMAT, 16, IER)
      CALL CHECK(IER)
      CALL WRBLK(NIRFIL, 16, ENHMAT, 2, IER)
      CALL CHECK(IER)
      CALL WRBLK(NIRFIL, 110, ENHMAT, 2, IER)
      CALL CHECK(IER)
      CALL WRBLK(NIRFIL, 112, ENHMAT, 16, IER)
      CALL CHECK(IER)

      TYPE"THRESHOLDED ENHANCED IMAGE COMPLETE"
      RETURN
      END
```

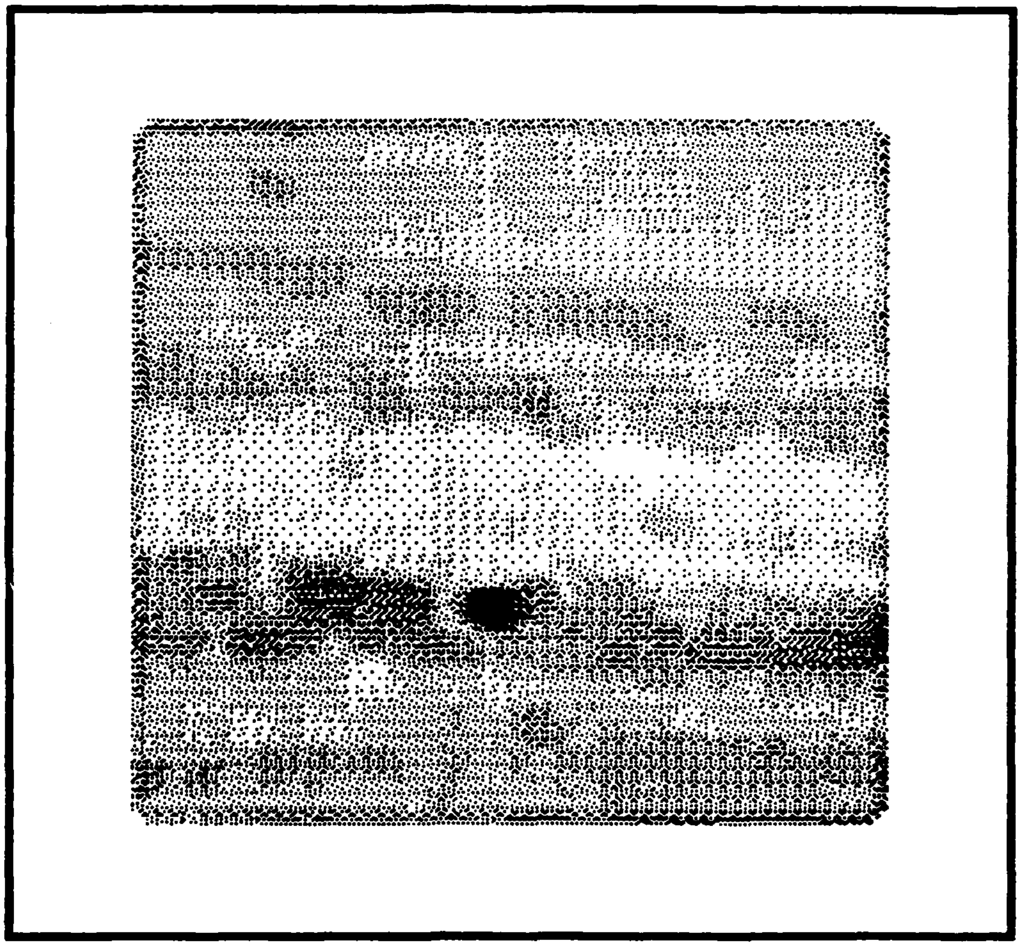
```
      SUBROUTINE NTROW1(K,MASKSZ,TOGGLE,NRSUM,PCT,DMIN,DMAX,
     +                  INMAT,OUTMAT)

      INTEGER K,MASKSZ,TOGGLE,OFF
      REAL NRSUM,PCT,DMIN,DMAX,INMAT(128,32),OUTMAT(128,16)
      REAL FCOLSM,CHANGE,ASUM,AVG,DSUM,DEV

      IF(TOGGLE.EQ.1)GO TO 1   ; INITIALIZE INDEXING VARIABLES
         K1=7
         GO TO 2
1        K1=23
2     CONTINUE
      OFF=(MASKSZ-1)/2

      DO 3 J=1,16   ; FILL FIRST/LAST 16 PIXELS WITH ZERO
         OUTMAT(J,K)=0.0
         OUTMAT(J+112,K)=0.0
3     CONTINUE

      CHANGE=0.0          ; FIND NEW MATRIX SUM
      DO 4 I=0,MASKSZ-1
         CHANGE=CHANGE+INMAT(17-OFF+I,MOD(K+OFF+K1,32)+1)
4     CONTINUE
      ASUM=NRSUM+CHANGE

      CHANGE=0.0         ; FIND NEW NEXT ROW SUM
      DO 5 I=0,MASKSZ-1
         CHANGE=CHANGE+INMAT(17-OFF+I,MOD(K-OFF+K1,32)+1)
5     CONTINUE
      NRSUM=ASUM-CHANGE

      FCOLSM=0.0          ; INITIALIZE FCOLSM
      DO 6 I=0,MASKSZ-1
         FCOLSM=FCOLSM+INMAT(17+OFF,MOD(K-OFF+I+K1,32)+1)
6     CONTINUE

      DO 10 J=17,112       ; COLUMN COUNT

         CHANGE=0.0       ; FIND NEW MATRIX SUM AND AVERAGE
         DO 7 I=0,MASKSZ-1
            CHANGE=CHANGE+INMAT(J+OFF,MOD(K-OFF+I+K1,32)+1)
7        CONTINUE
         ASUM=ASUM+CHANGE-FCOLSM
         AVG=ASUM/MASKSZ**2

         FCOLSM=0.0       ; FIND NEW FIRST COLUMN SUM
         DO 8 I=0,MASKSZ-1
            FCOLSM=FCOLSM+INMAT(J-OFF,MOD(K-OFF+I+K1,32)+1)
8        CONTINUE

         DSUM=0.0    ; FIND NEW DEVIATION
         DO 9 I1=0,MASKSZ-1
            DO 9 I2=0,MASKSZ-1
               DSUM=DSUM+IABS(INMAT(J-OFF+I1,MOD(K-OFF+I2+K1,32)+1)-AVG)
9        CONTINUE
         DEV=DSUM/MASKSZ**2

         DEV=DEV*PCT                ; MODIFY DEVIATION
         IF(DEV.LT.DMIN)DEV=DMIN
         IF(DEV.GT.DMAX)DEV=DMAX
         OUTMAT(J,K)=0.0             ; THRESHOLD RULE
         IF(INMAT(J,MOD(K+K1,32)+1).GT.(AVG+DEV))OUTMAT(J,K)=15.0

10    CONTINUE

      RETURN
      END
```

APPENDIX E

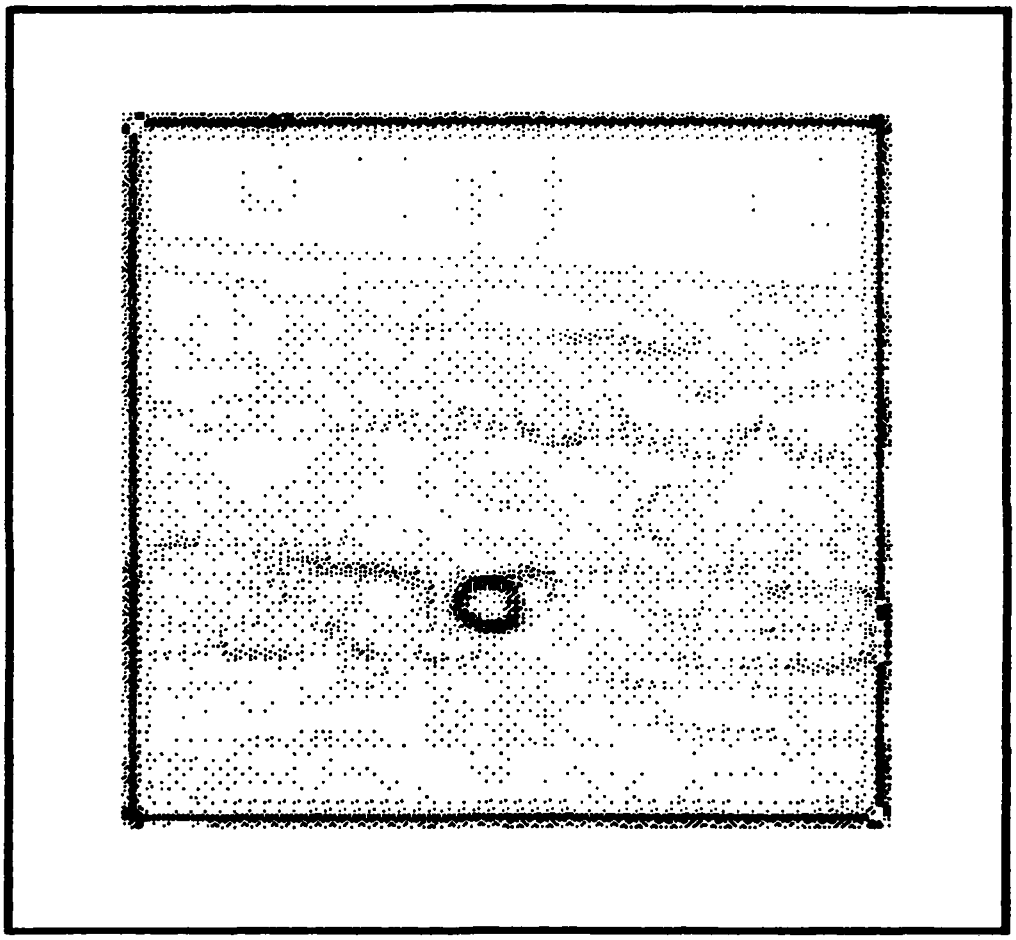
MORE CLUSTER RECOGNITION RESULTS



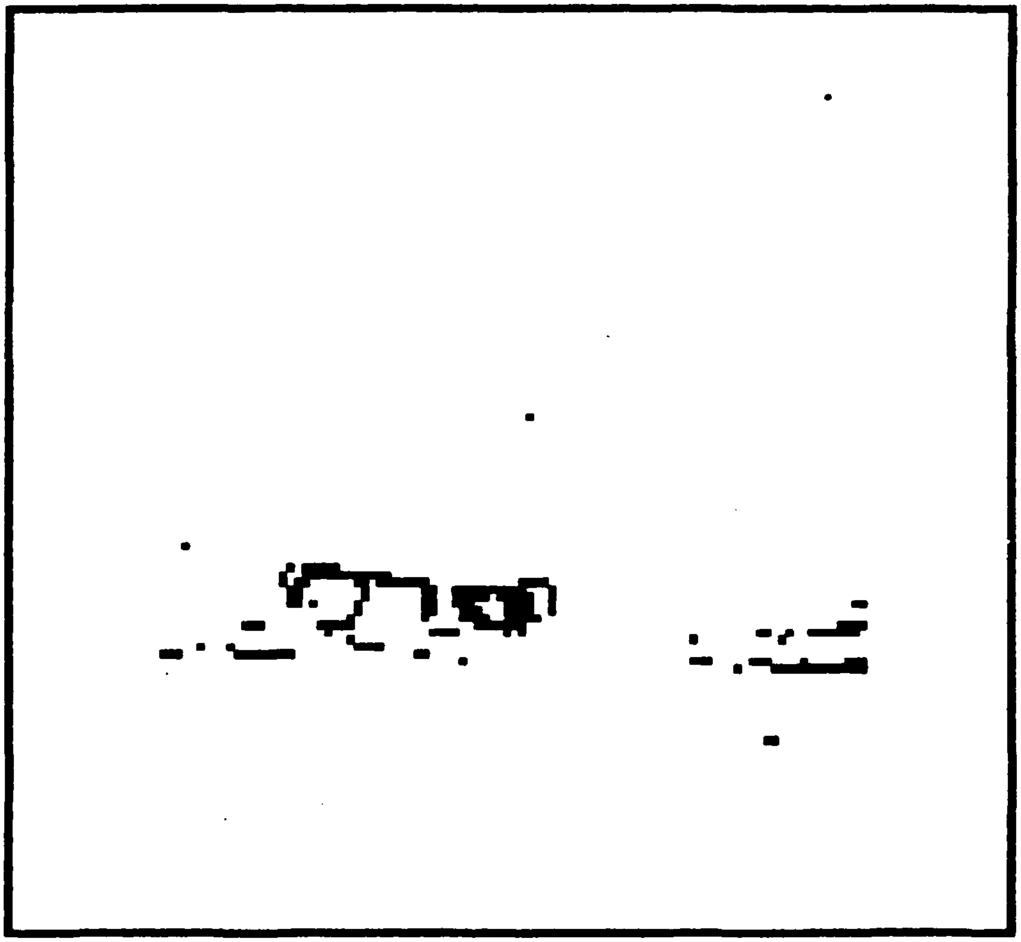
INFRARED IMAGE #2 (IMAG2.IR)



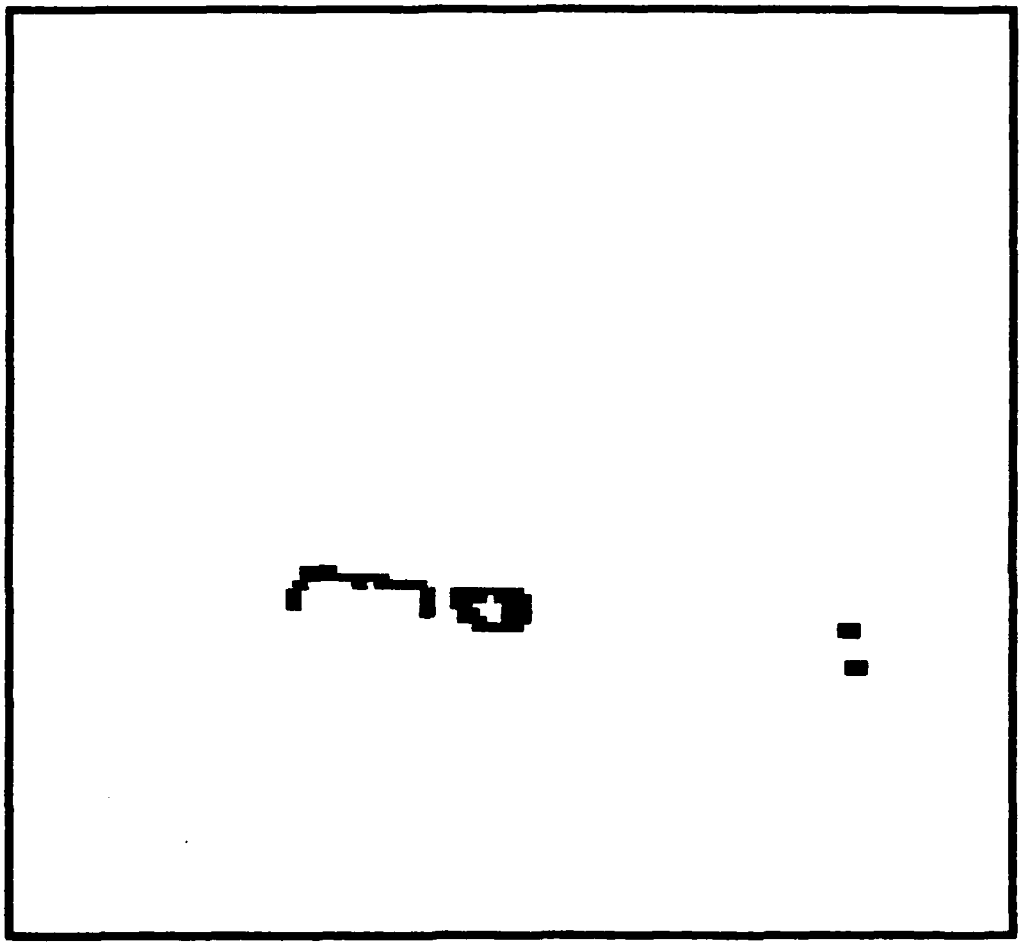
ENHANCED IMAGE OF IMAG2.IR



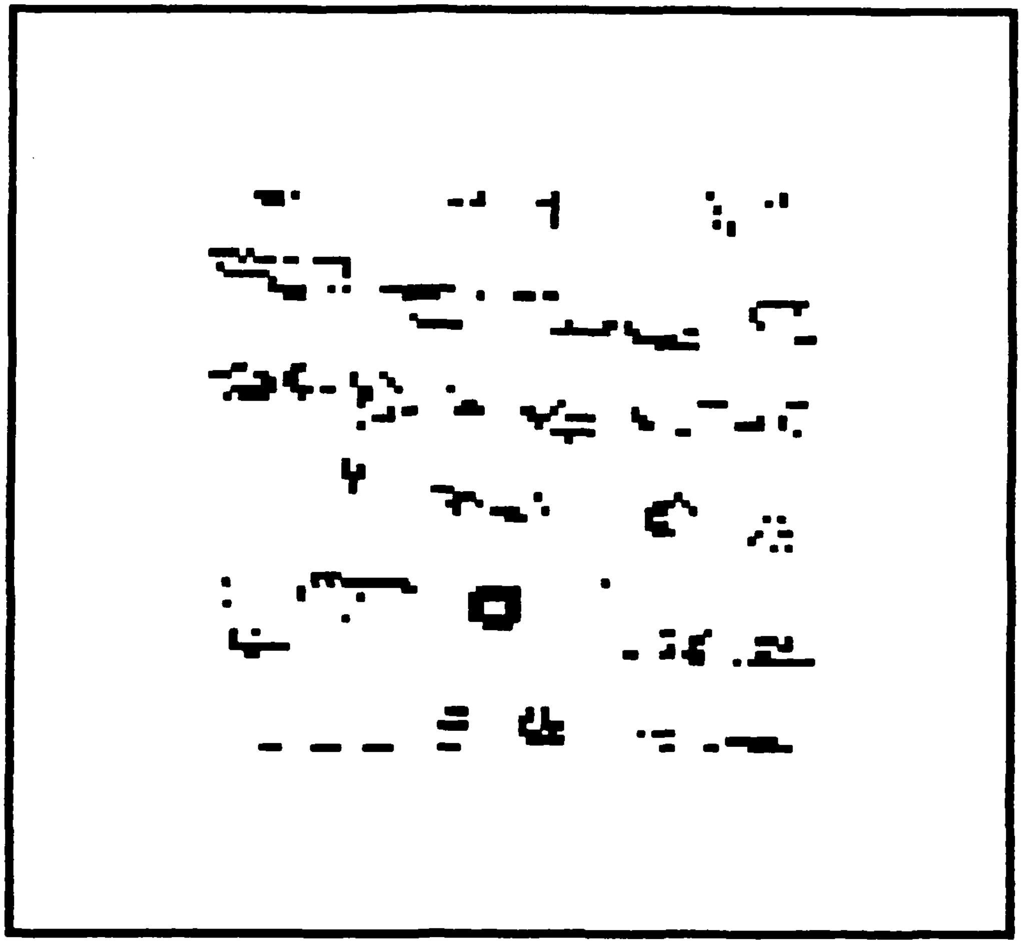
EDGED IMAGE OF IMAG2.IR



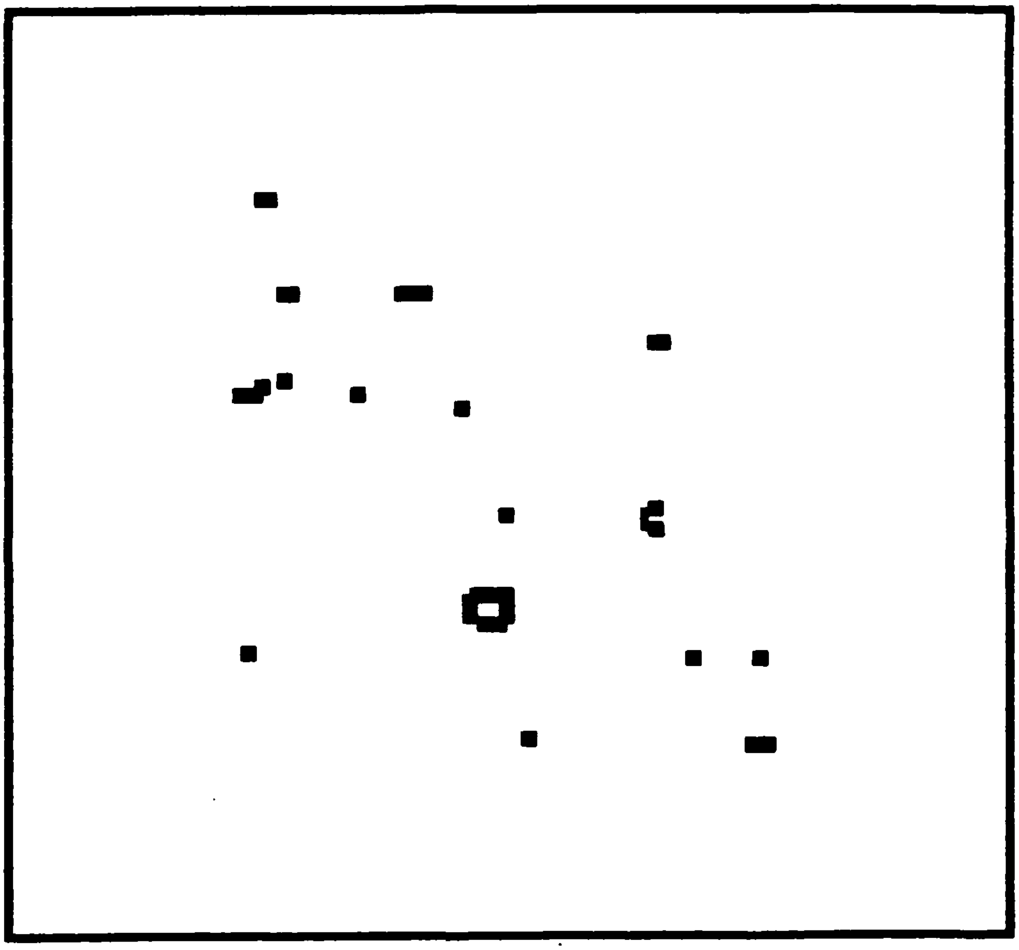
GLOBALLY THRESHOLDED IMAGE OF IMAG2. IR



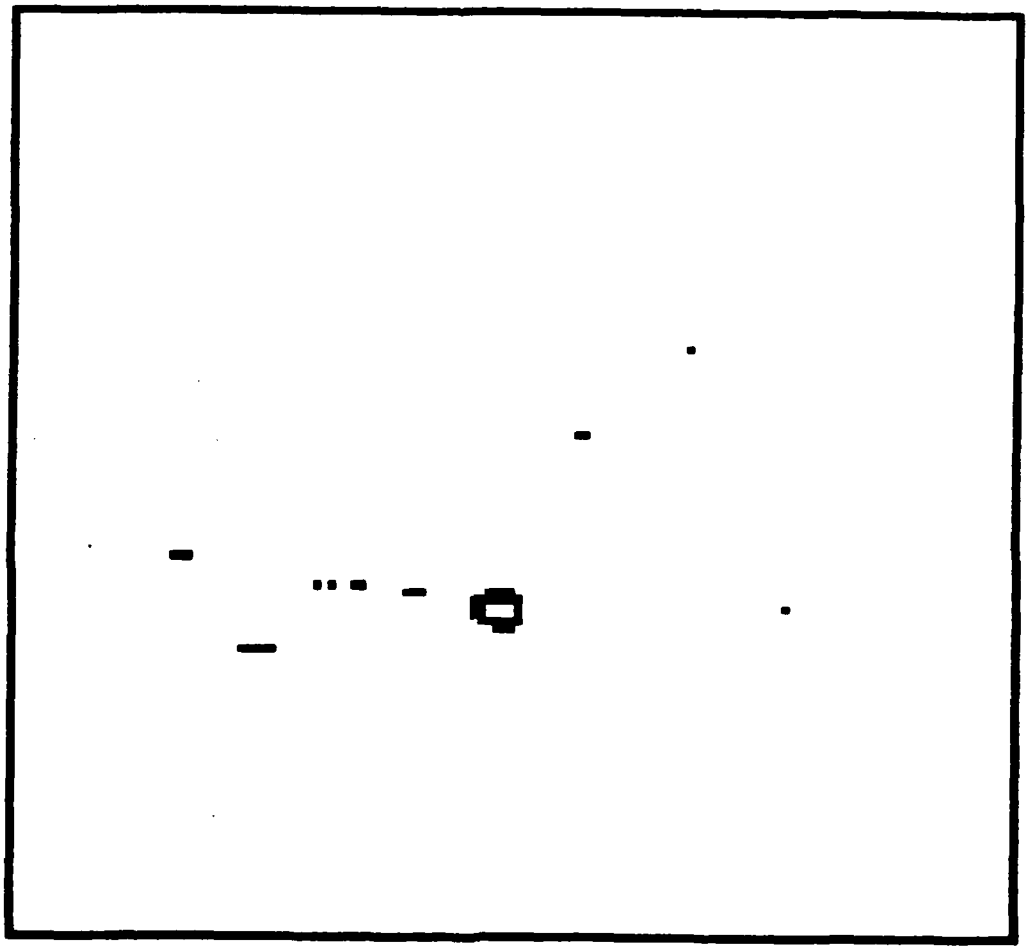
GLOBALLY THRESHOLDED IMAG2.IR AFTER CONNECTIVITY TEST



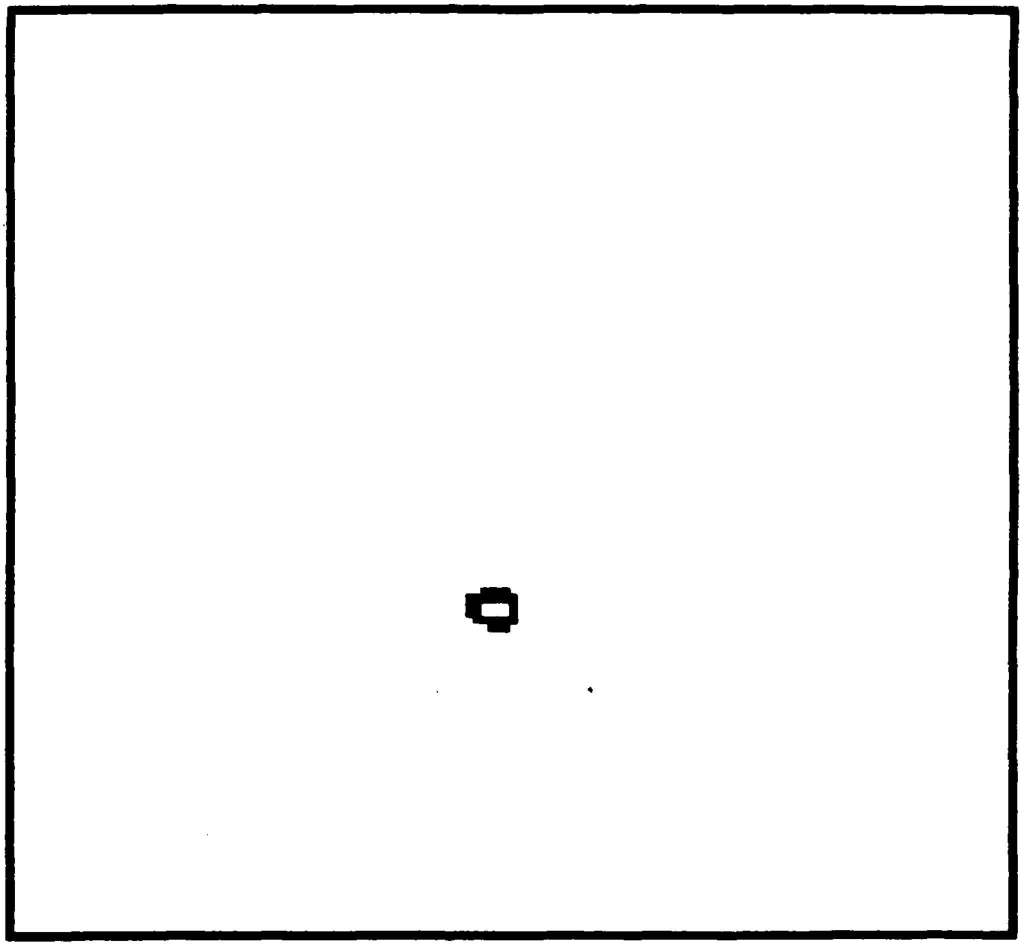
LOCALLY THRESHOLDED IMAG2.IR (17 X 17 NEIGHBORHOOD)



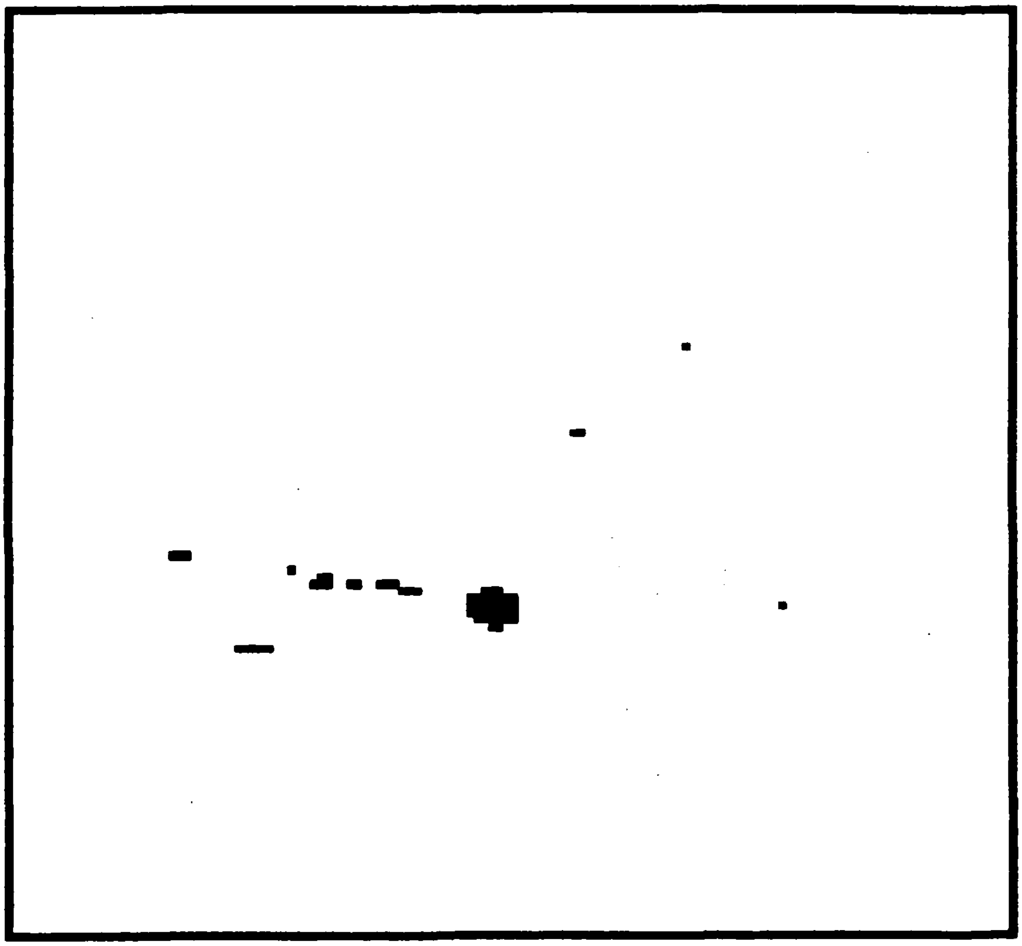
LOCALLY THRESHOLDED IMA02. IR (17 X 17 NEIGHBORHOOD)
AFTER CONNECTIVITY TEST



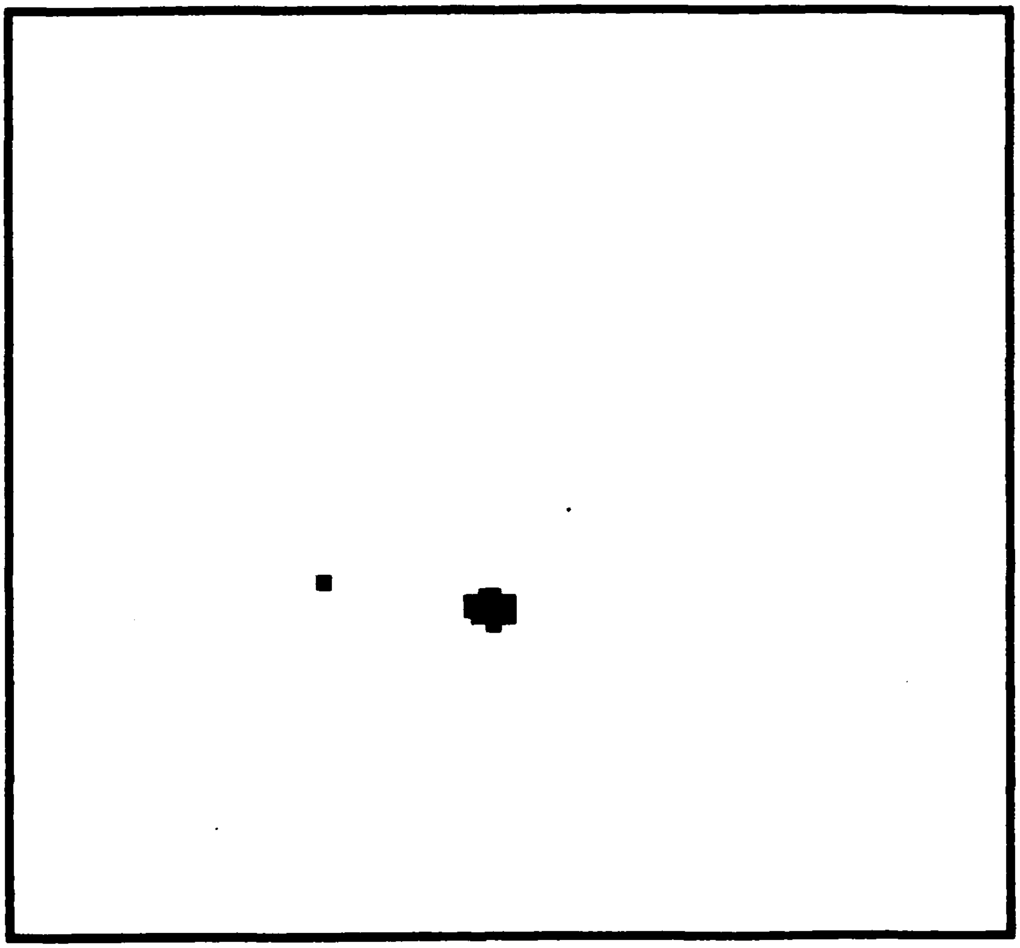
LOCALLY THRESHOLDED IMAG2.IR (7 X 7 NEIGHBORHOOD)



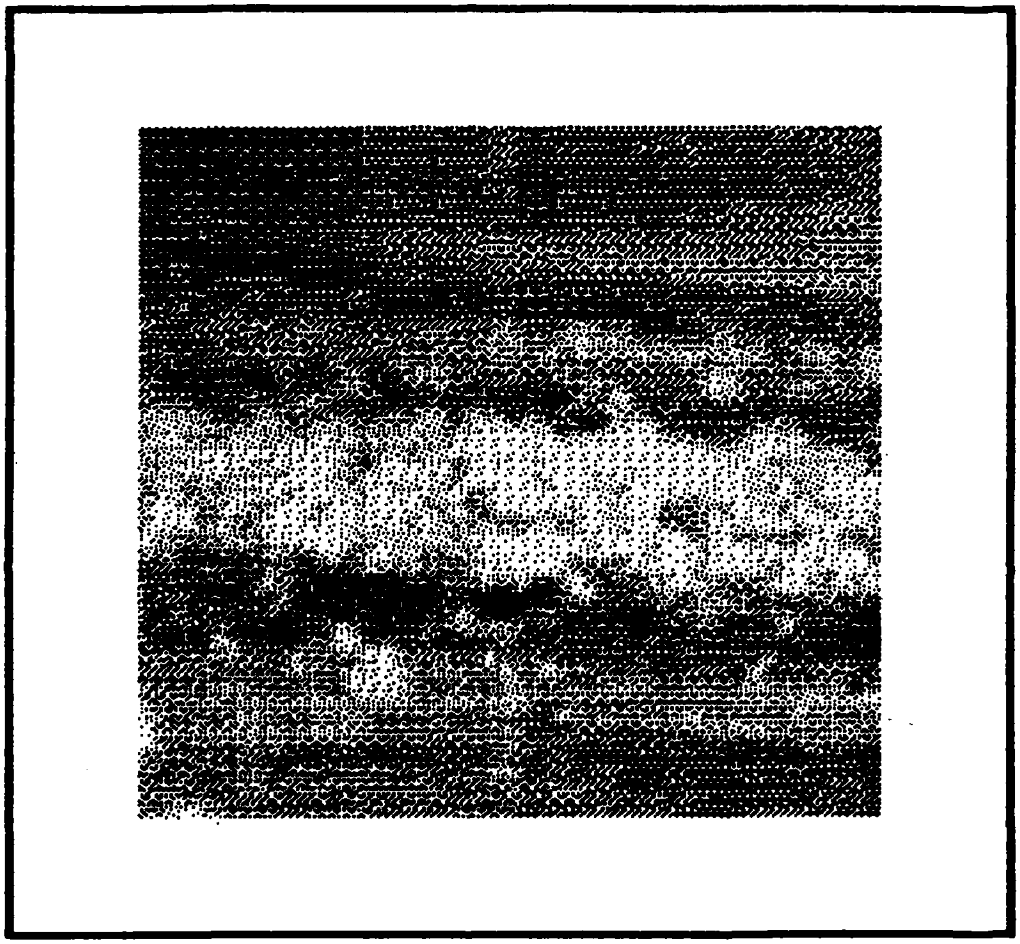
LOCALLY THRESHOLDED IMA02.IR (7 X 7 NEIGHBORHOOD)
AFTER CONNECTIVITY TEST



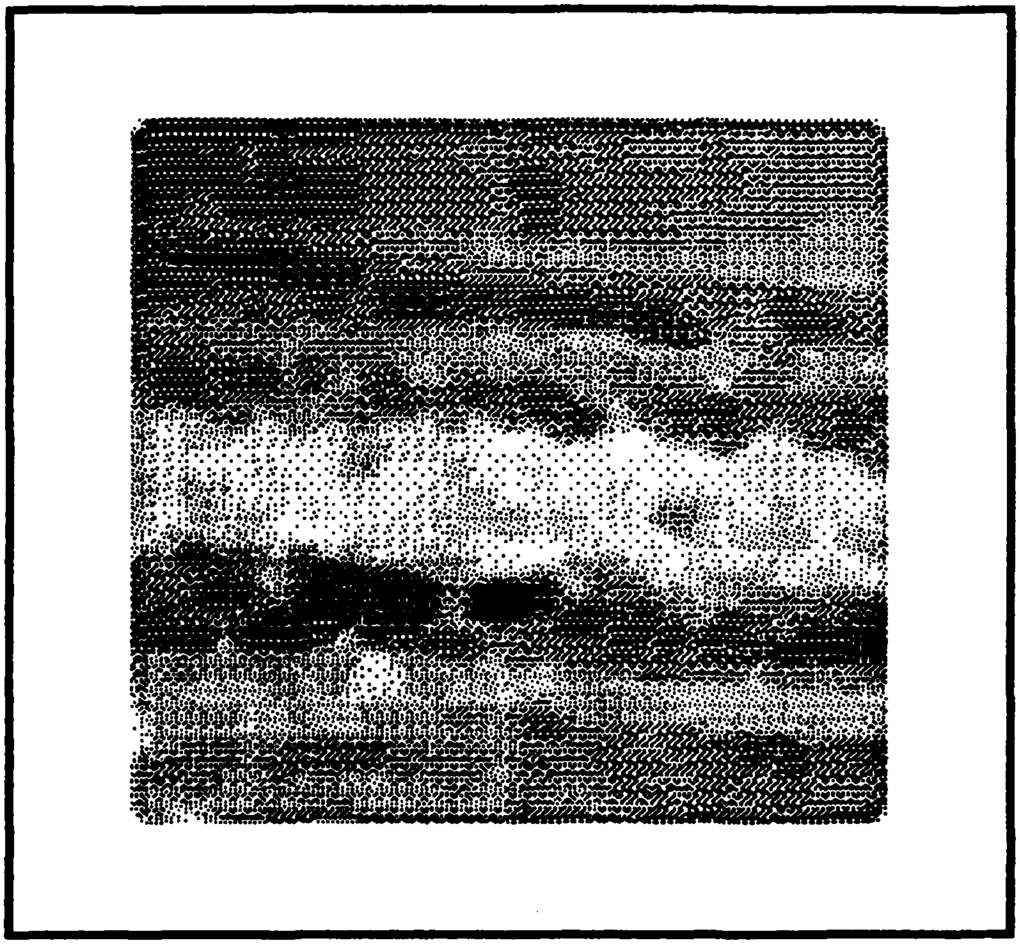
LOCALLY THRESHOLDED IMAG2.IR (USING 7 X 7 NEIGHBORHOOD WITHOUT EDGING)



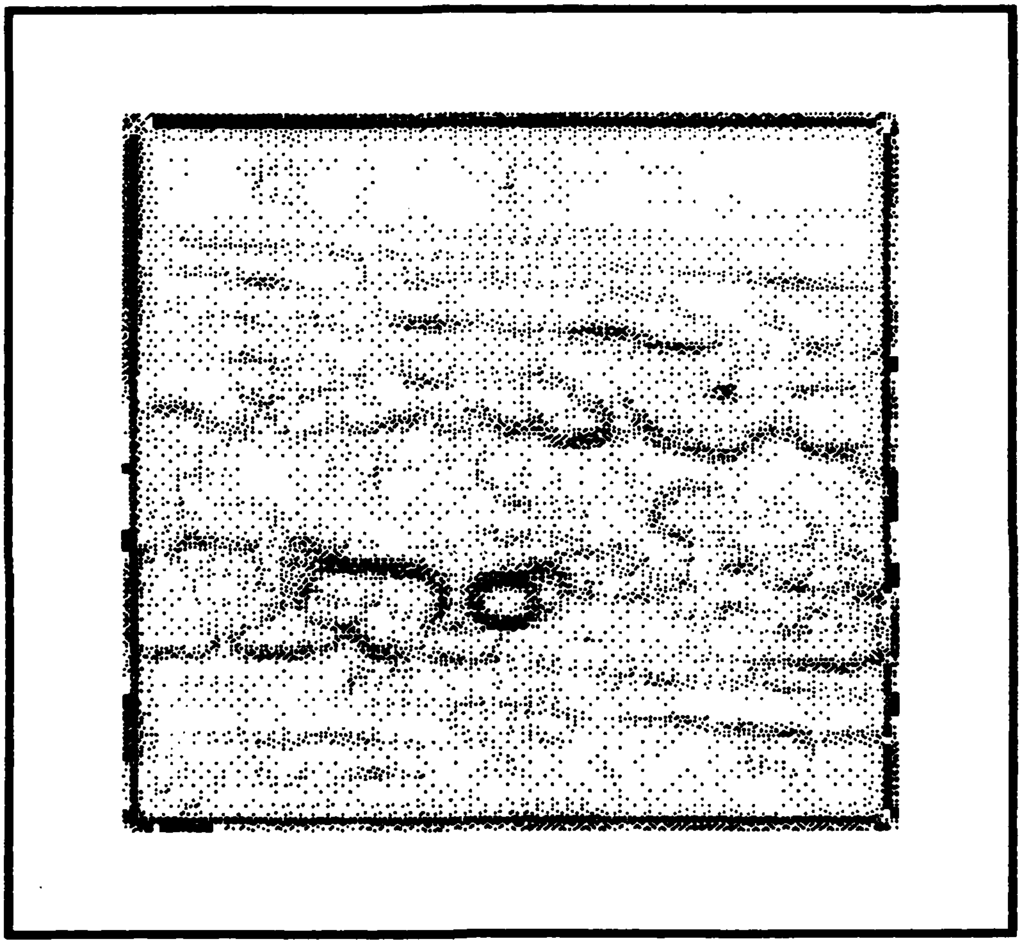
LOCALLY THRESHOLDED IMAG2.IR (USING 7 X 7 NEIGHBORHOOD
WITHOUT EDGING) AFTER CONNECTIVITY TEST



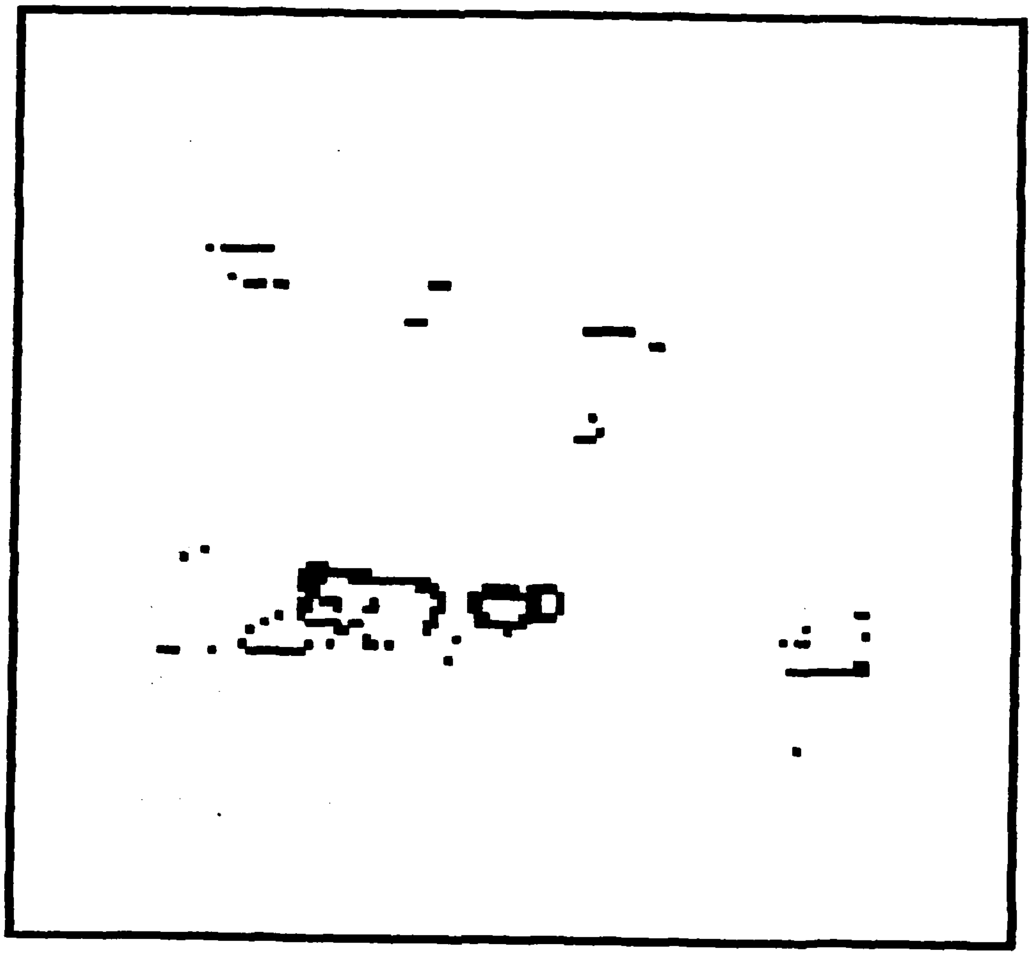
INFRARED IMAGE #3 (IMAG3.IR)



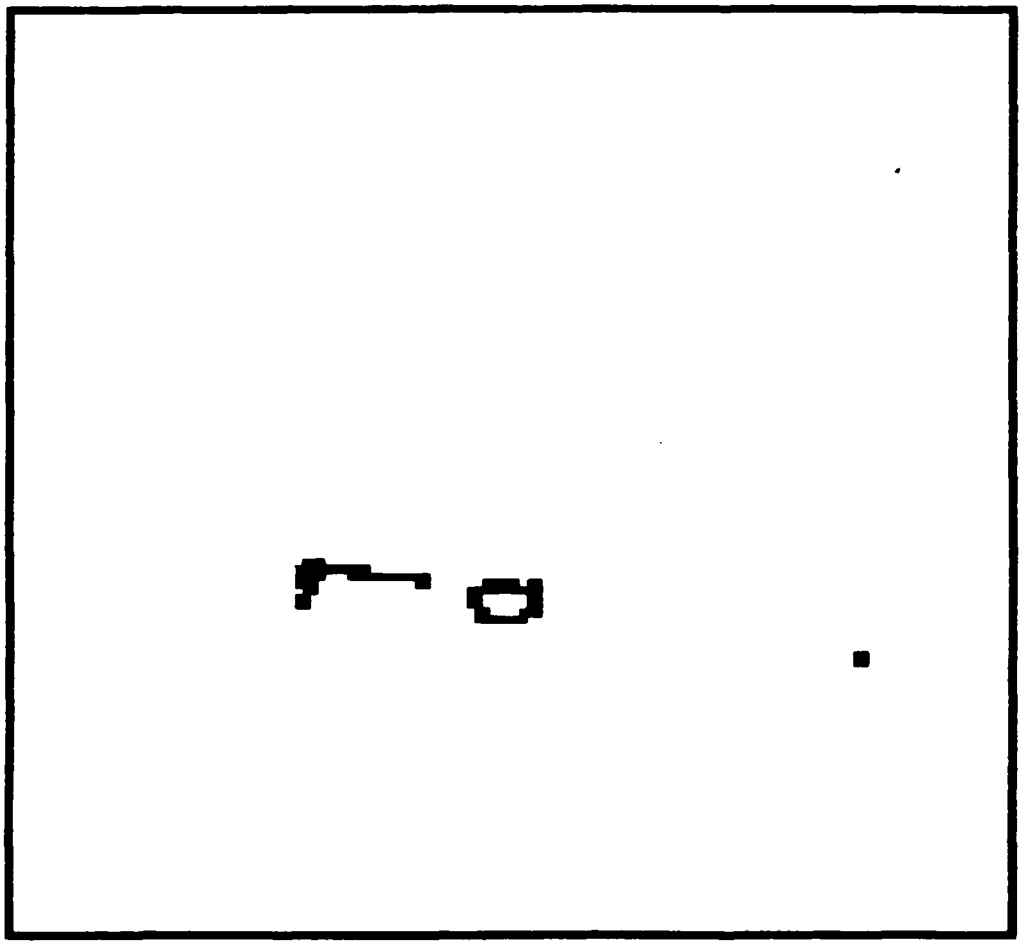
ENHANCED IMAGE OF IMAG3. IR



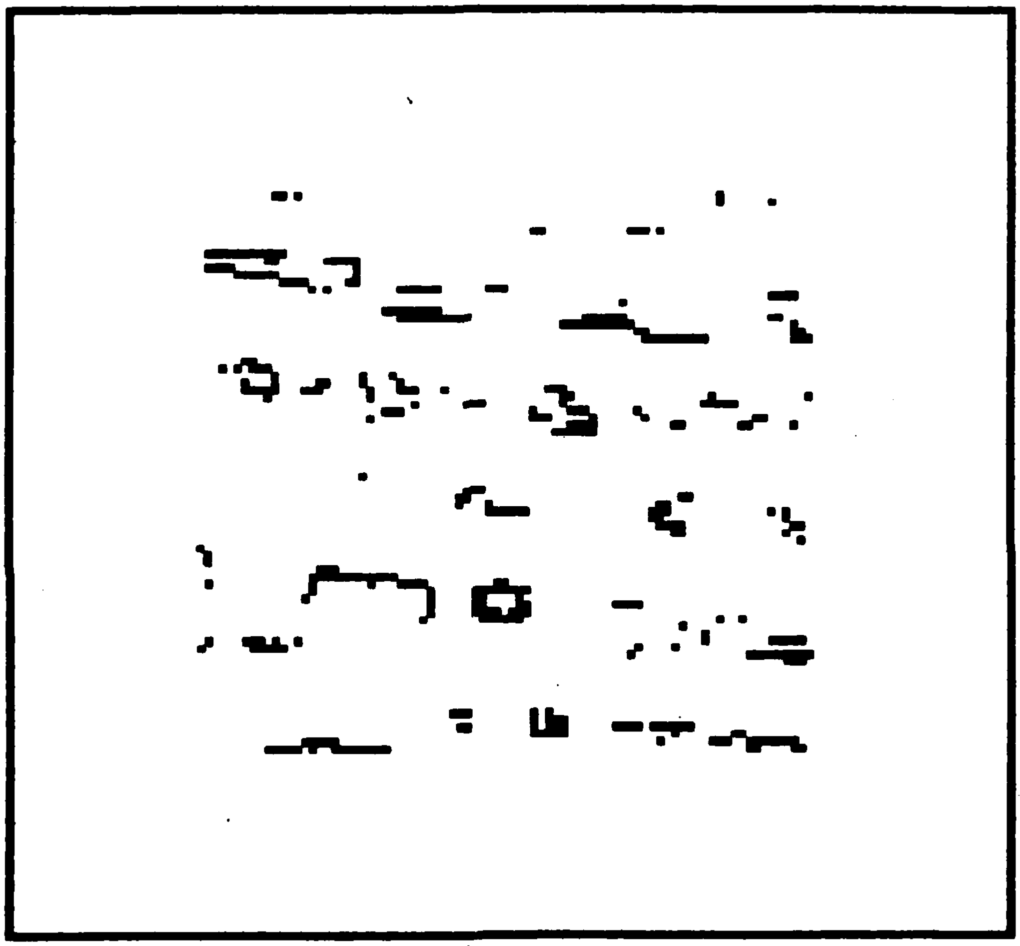
EDGED IMAGE OF IMAG3. IR



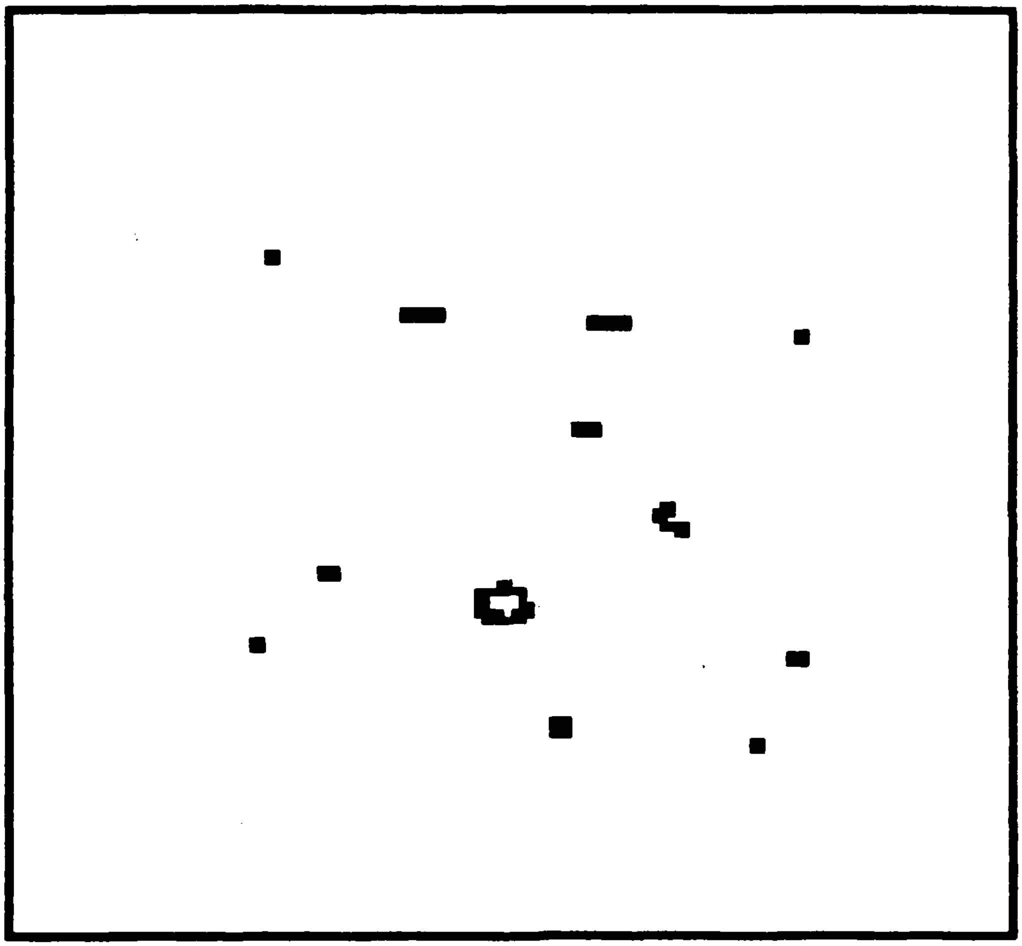
GLOBALLY THRESHOLDED IMAGE OF IMAG3.IR



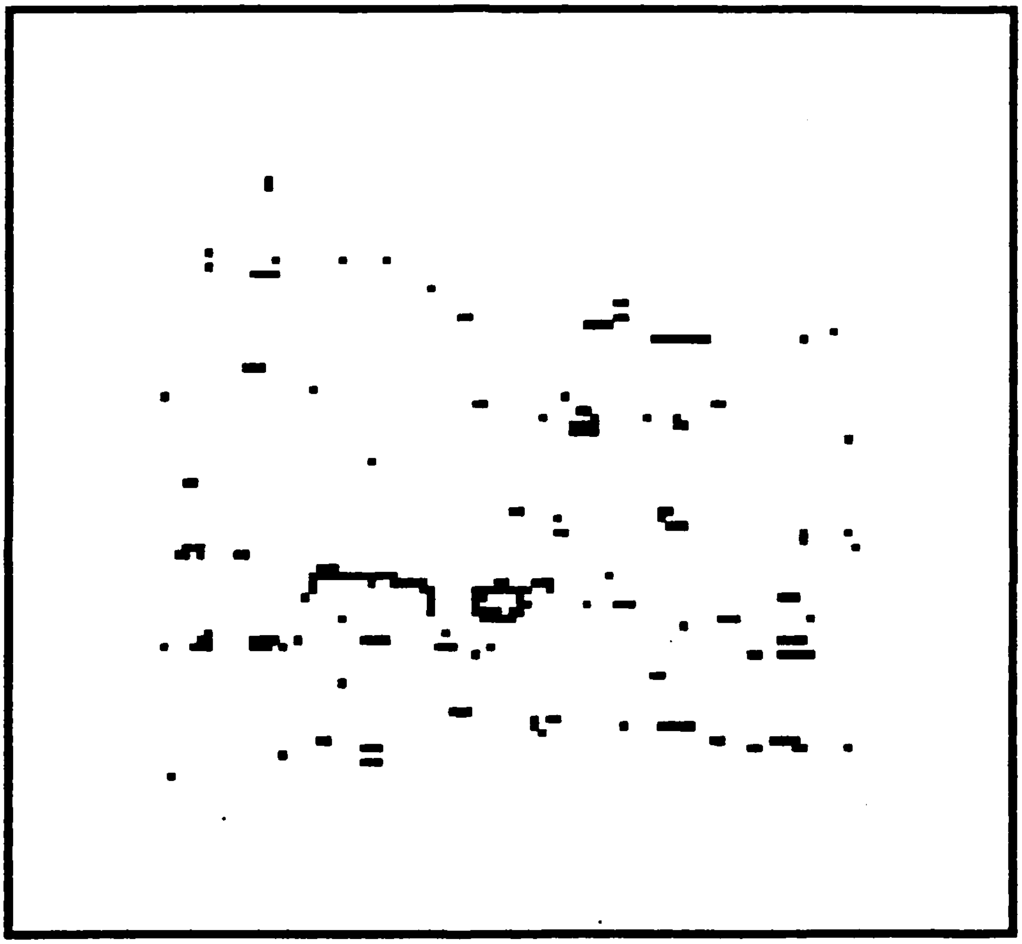
GLOBALLY THRESHOLDED IMAG3.IR AFTER CONNECTIVITY TEST



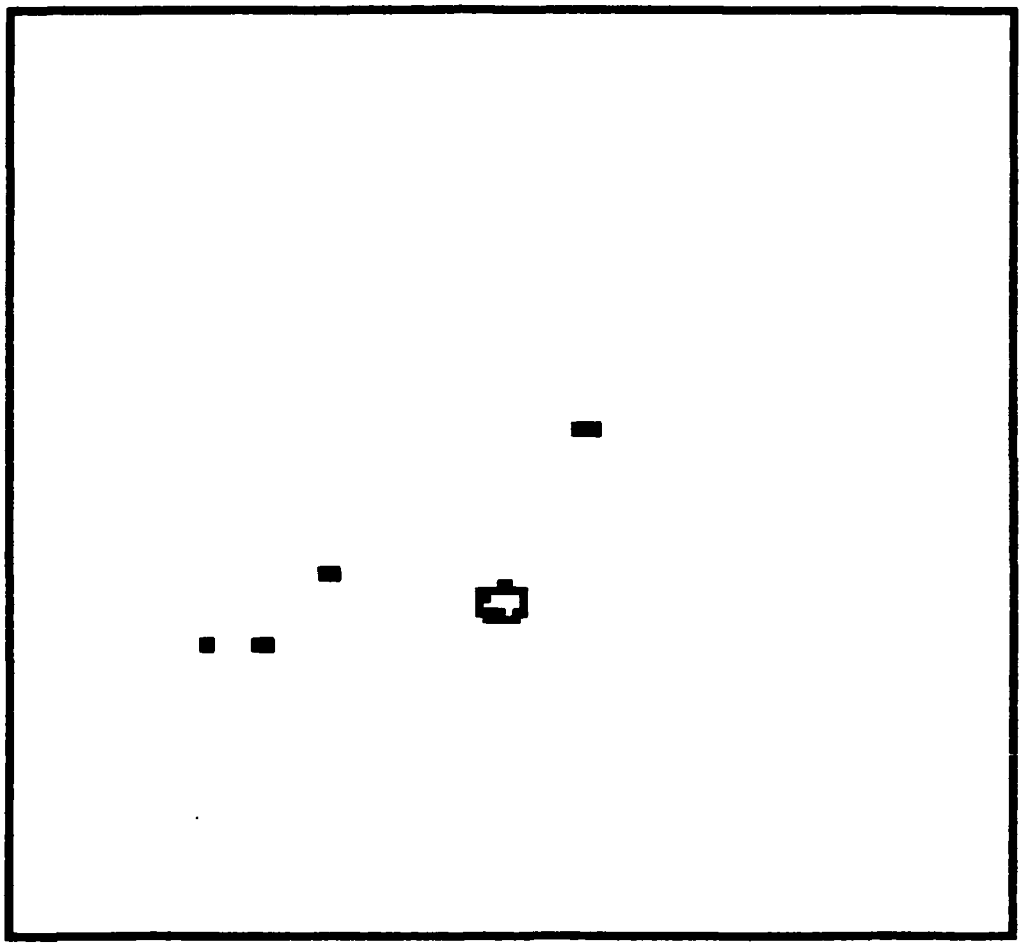
LOCALLY THRESHOLDED IMAG3.IR (17 X 17 NEIGHBORHOOD)



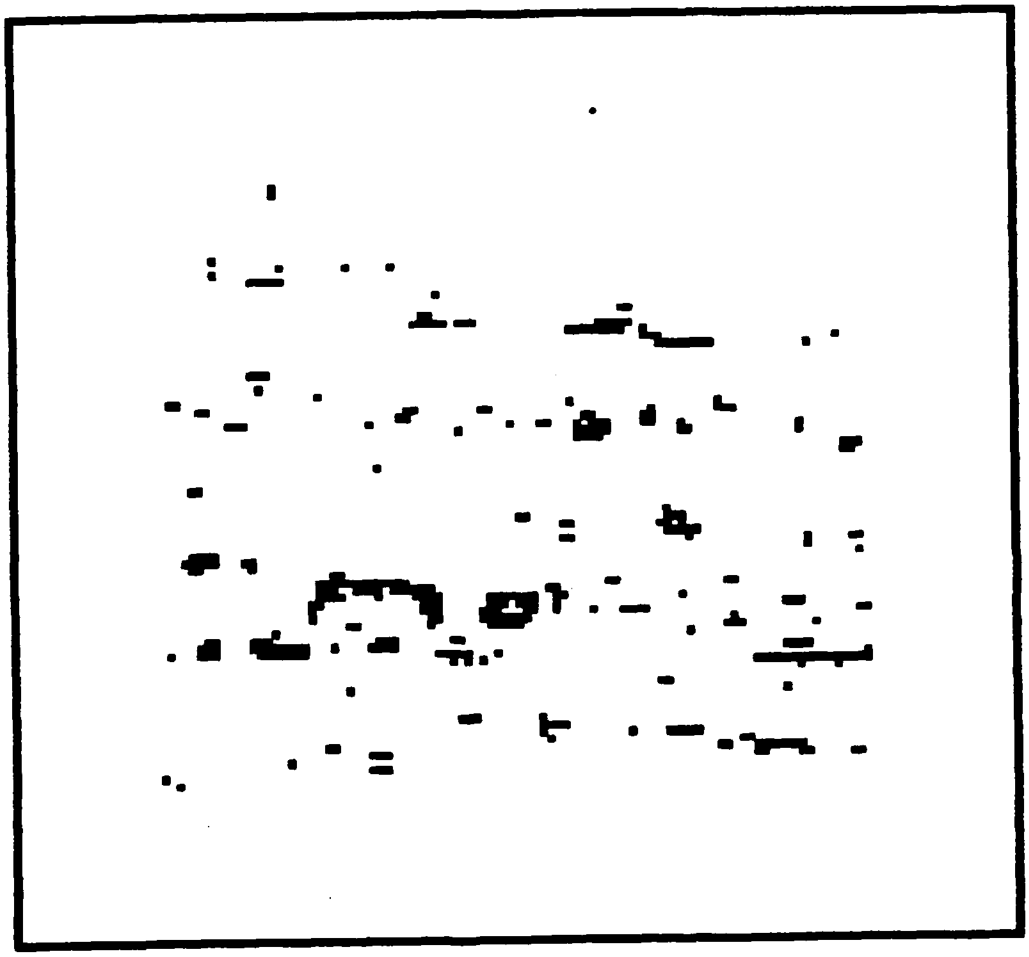
LOCALLY THRESHOLDED IMAQ3.IR (17 X 17 NEIGHBORHOOD)
AFTER CONNECTIVITY TEST



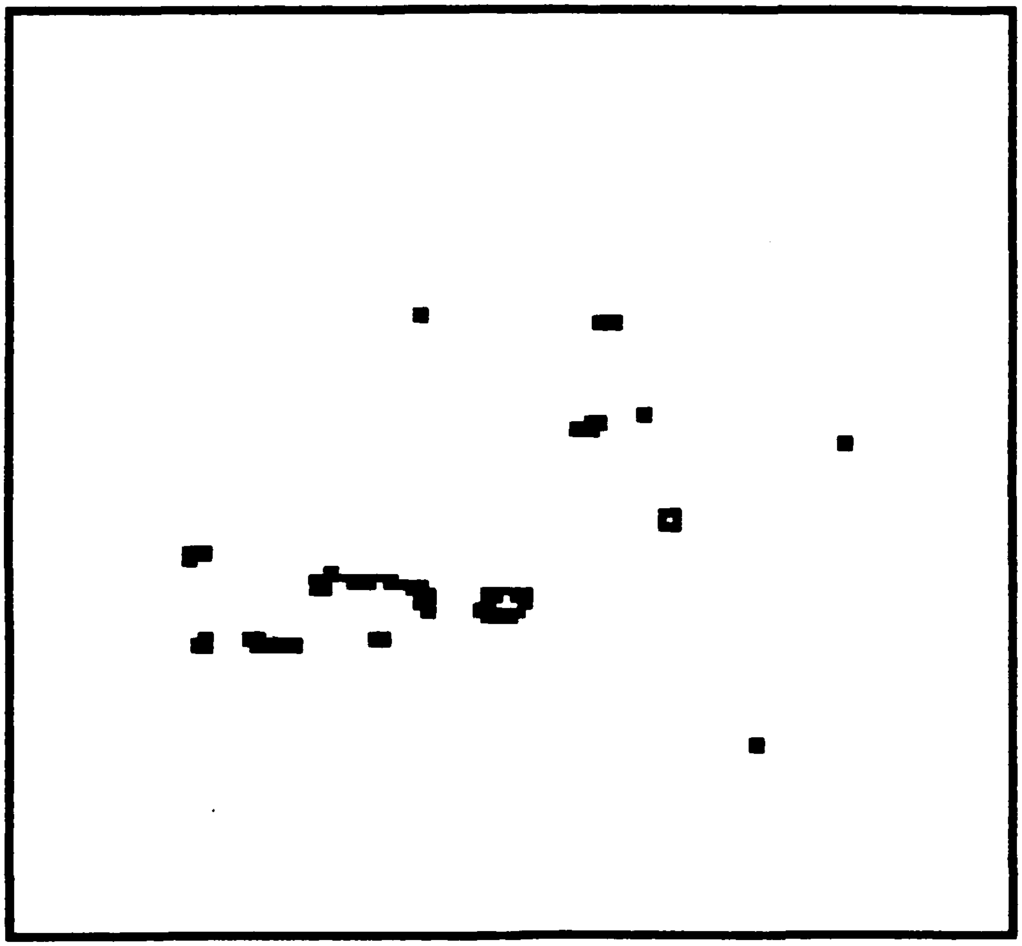
LOCALLY THRESHOLDED IMAG3.IR (7 X 7 NEIGHBORHOOD)



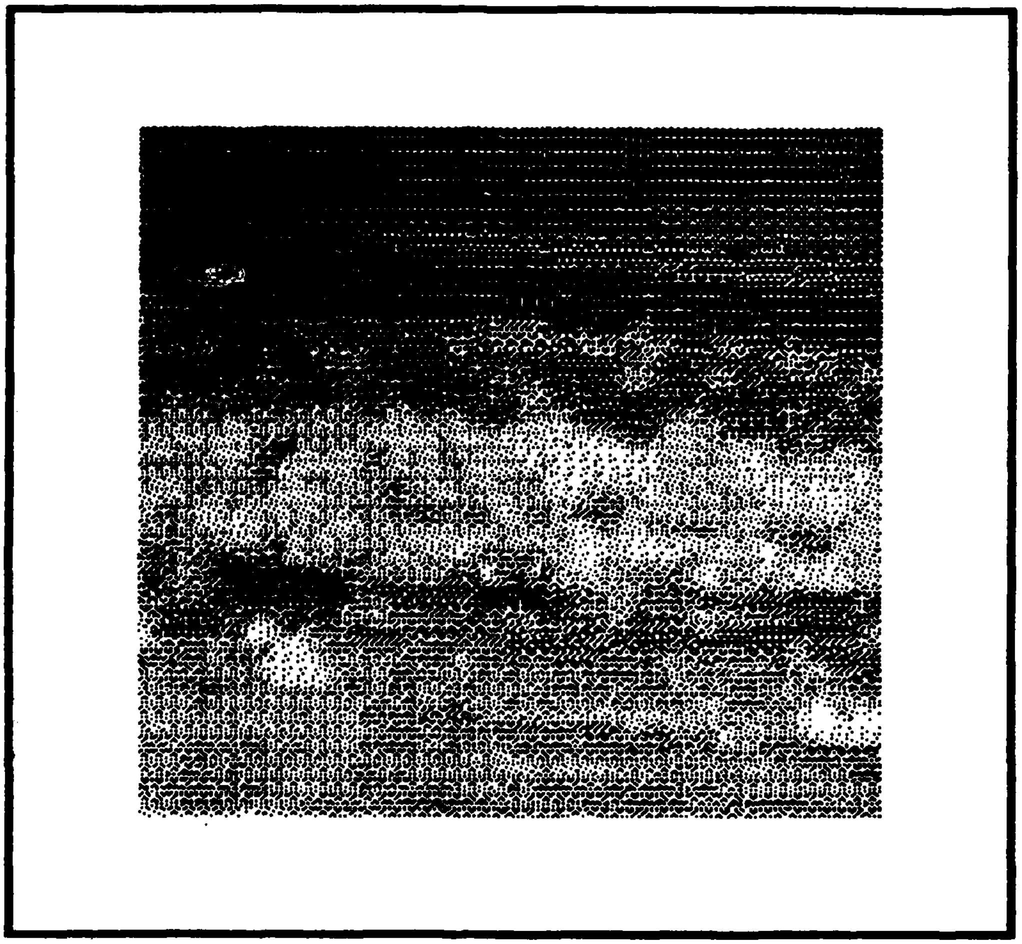
LOCALLY THRESHOLDED IMAG3.IR (7 X 7 NEIGHBORHOOD)
AFTER CONNECTIVITY TEST



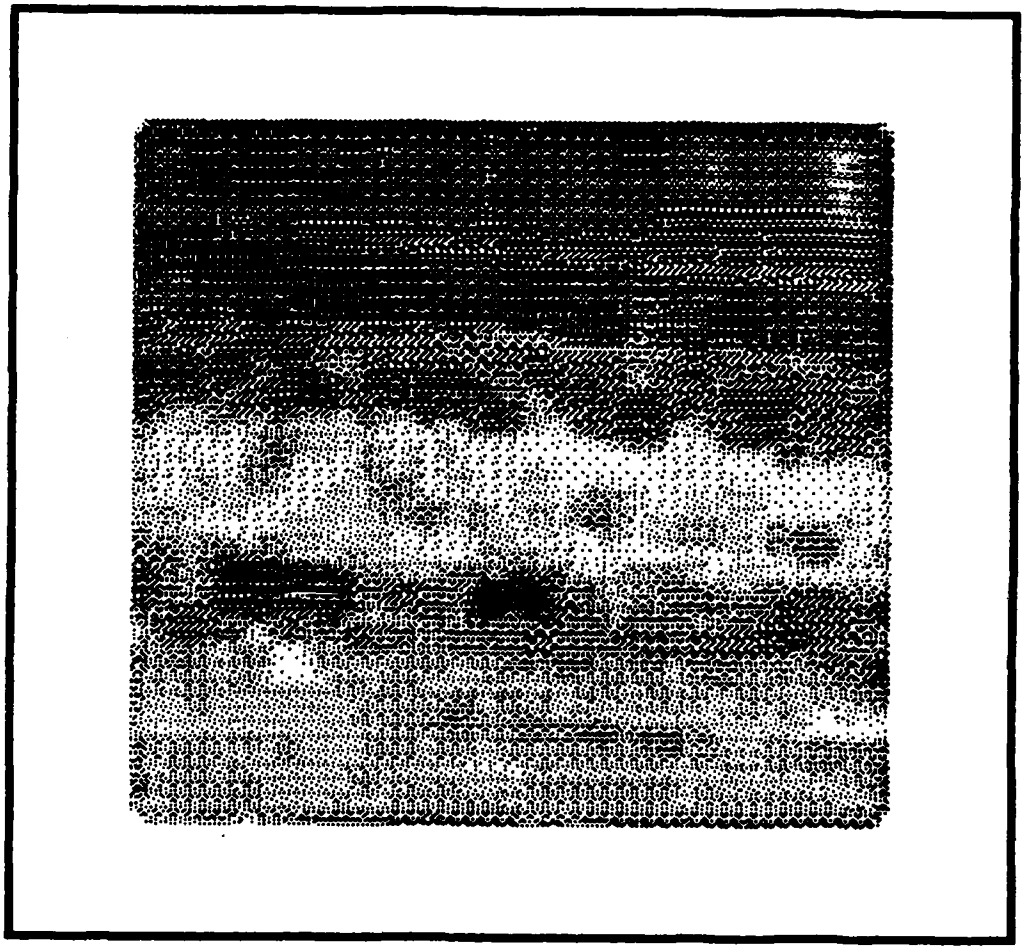
LOCALLY THRESHOLDED IMAG3.IR (USING 7 X 7 NEIGHBORHOOD WITHOUT EDGING)



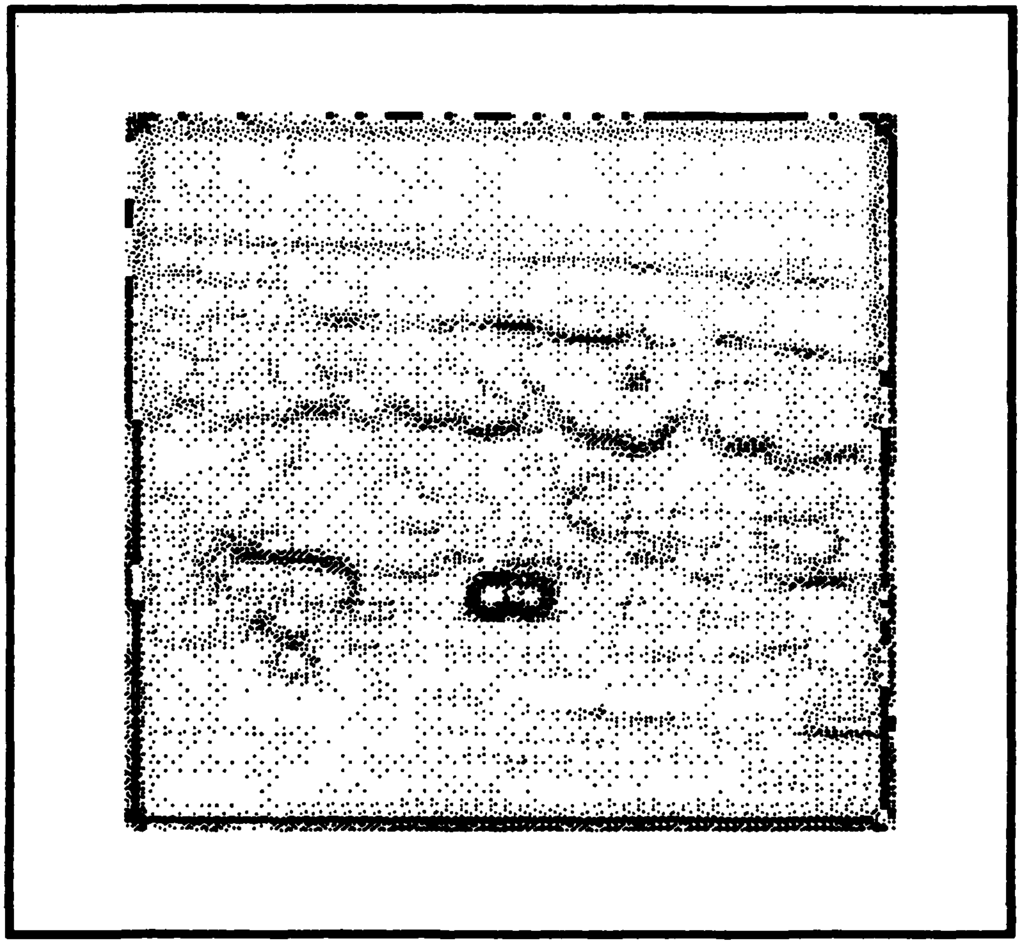
LOCALLY THRESHOLDED IMAG3.IR (USING 7 X 7 NEIGHBORHOOD
WITHOUT EDGING) AFTER CONNECTIVITY TEST



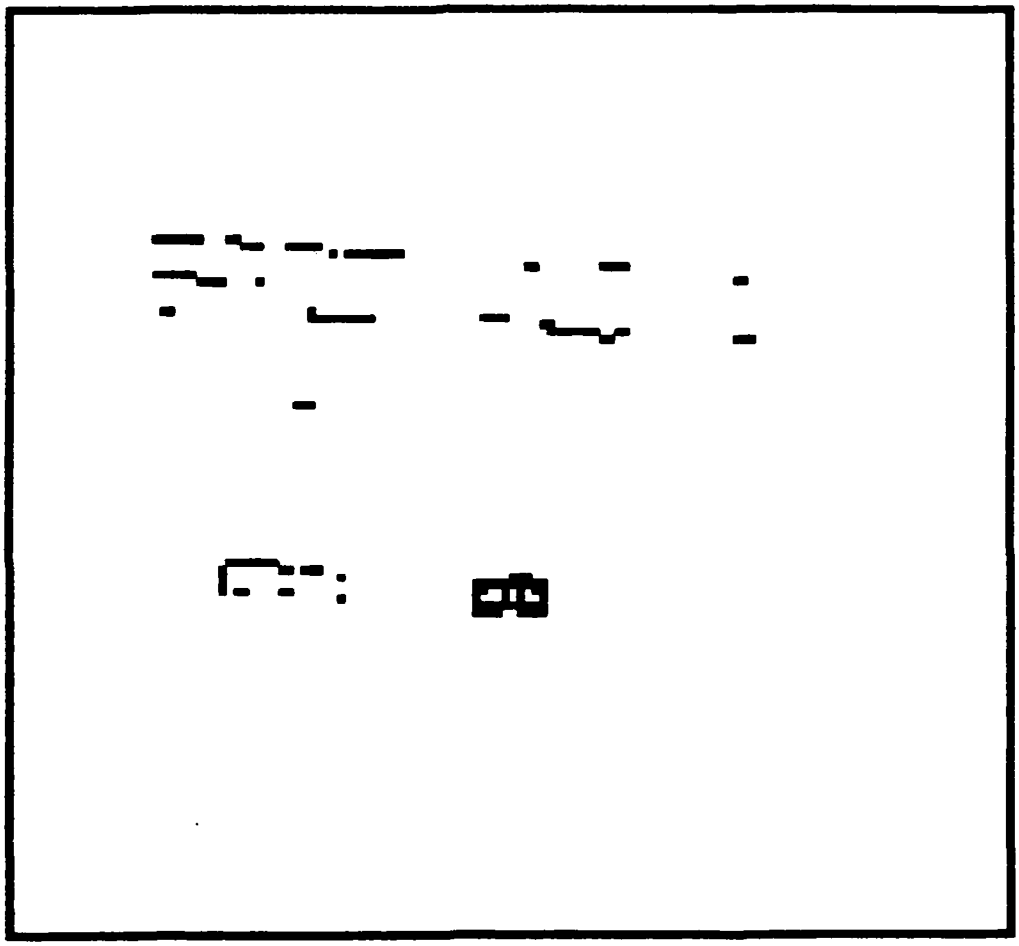
INFRARED IMAGE #4 (IMAG4.IR)



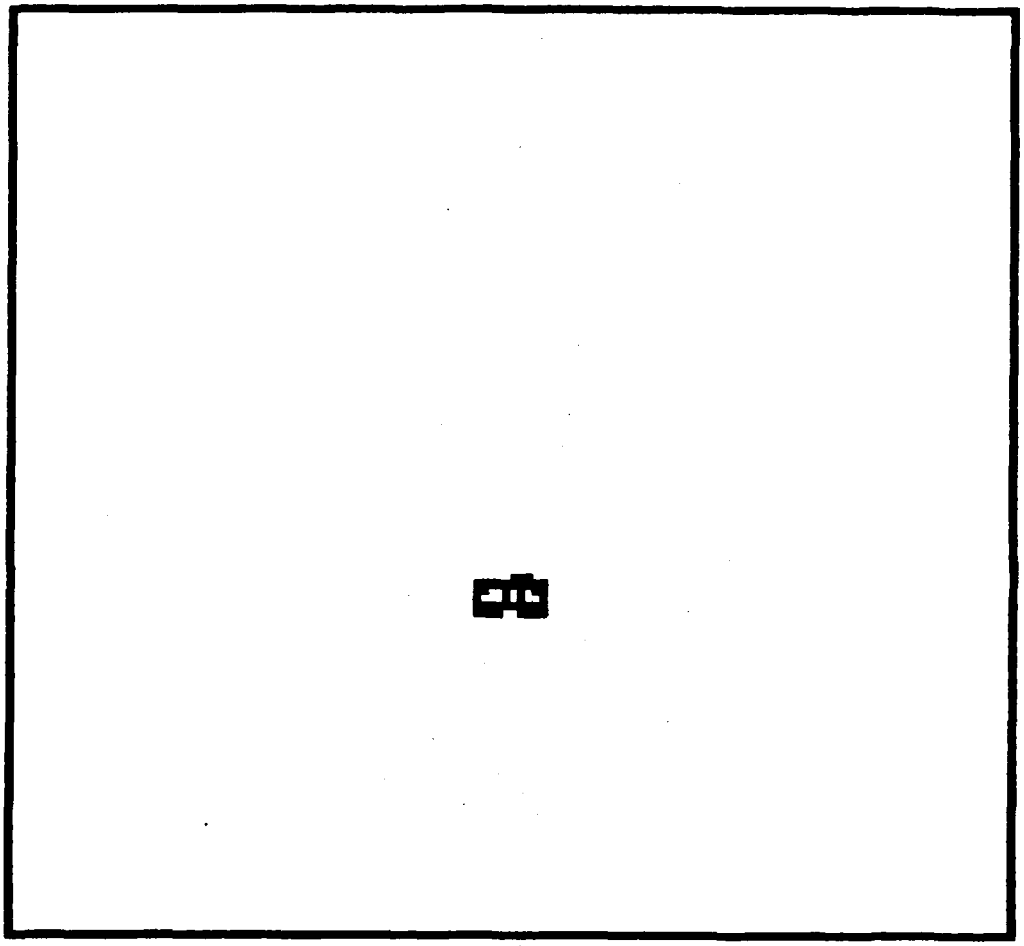
ENHANCED IMAGE OF IMAQ4. IR



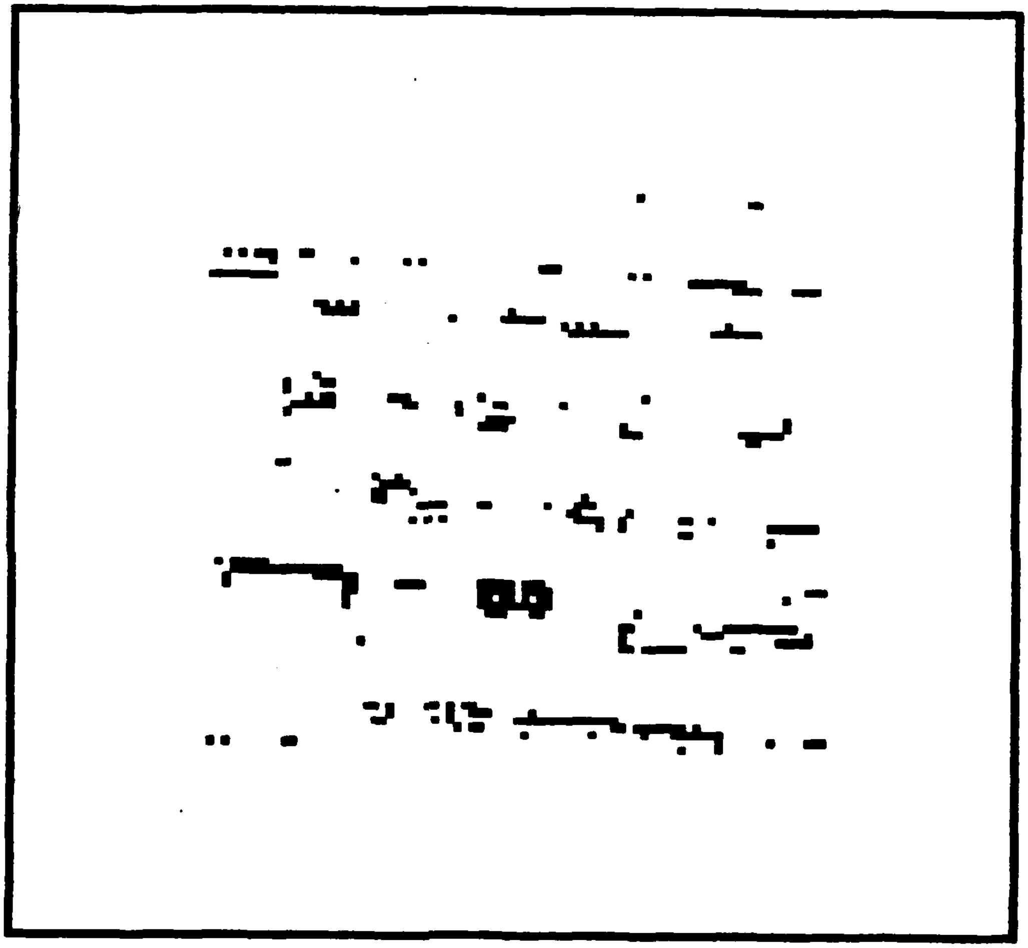
EDGED IMAGE OF IMAG4.IR



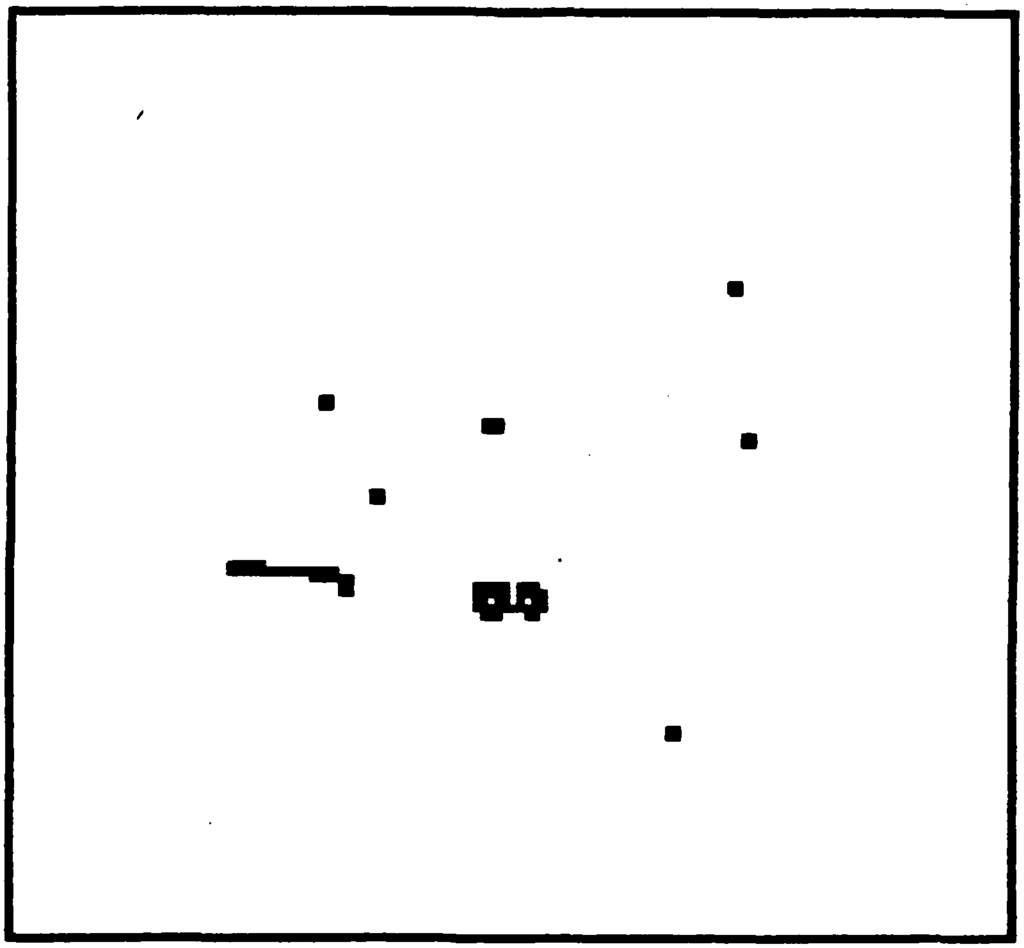
GLOBALLY THRESHOLDED IMAGE OF IMAG4. IR



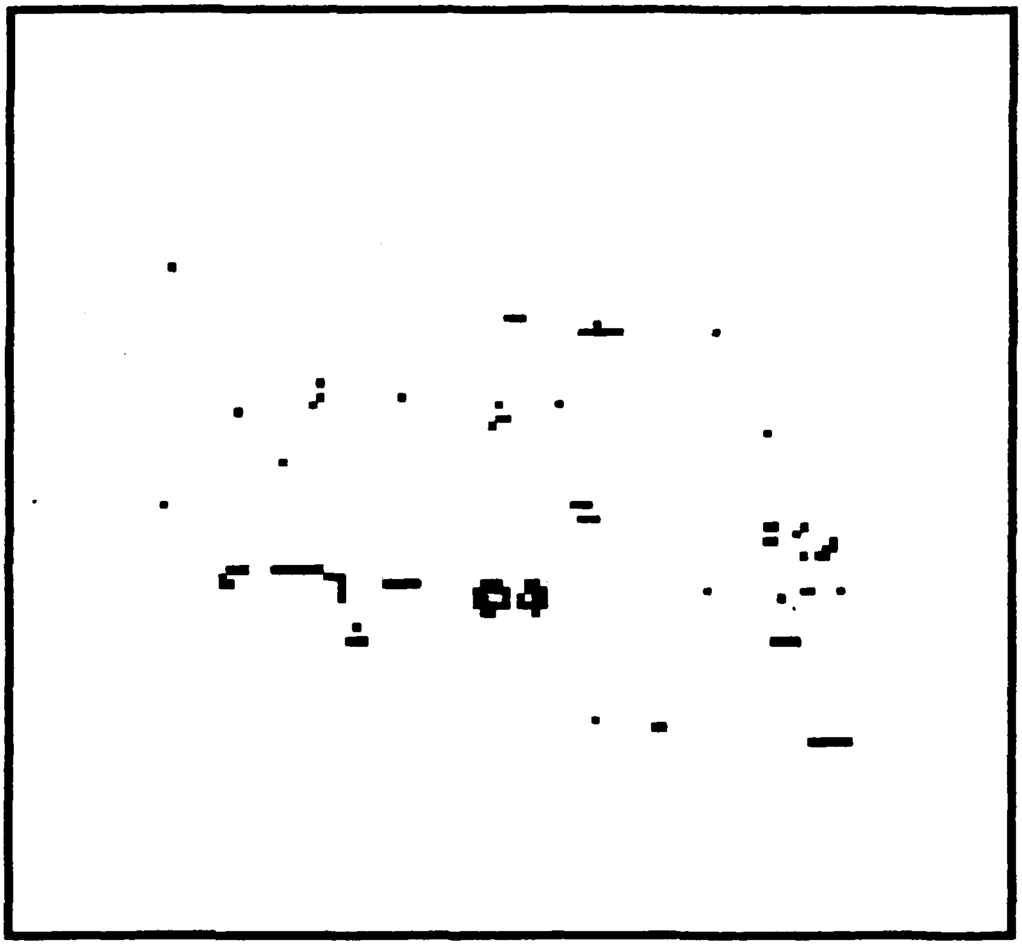
GLOBALLY THRESHOLDED IMAG4.IR AFTER CONNECTIVITY TEST



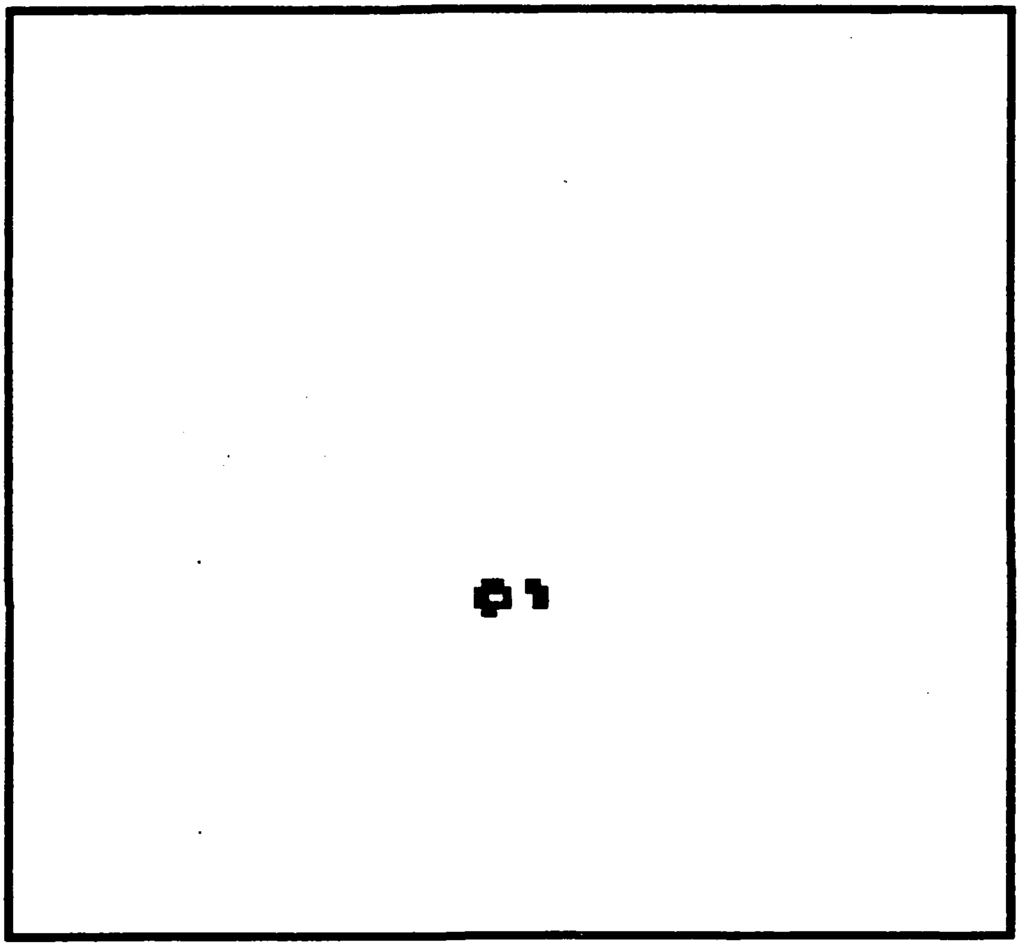
LOCALLY THRESHOLDED IMAG4.IR (17 X 17 NEIGHBORHOOD)



LOCALLY THRESHOLDED IMAG4. IR (17 X 17 NEIGHBORHOOD)
AFTER CONNECTIVITY TEST

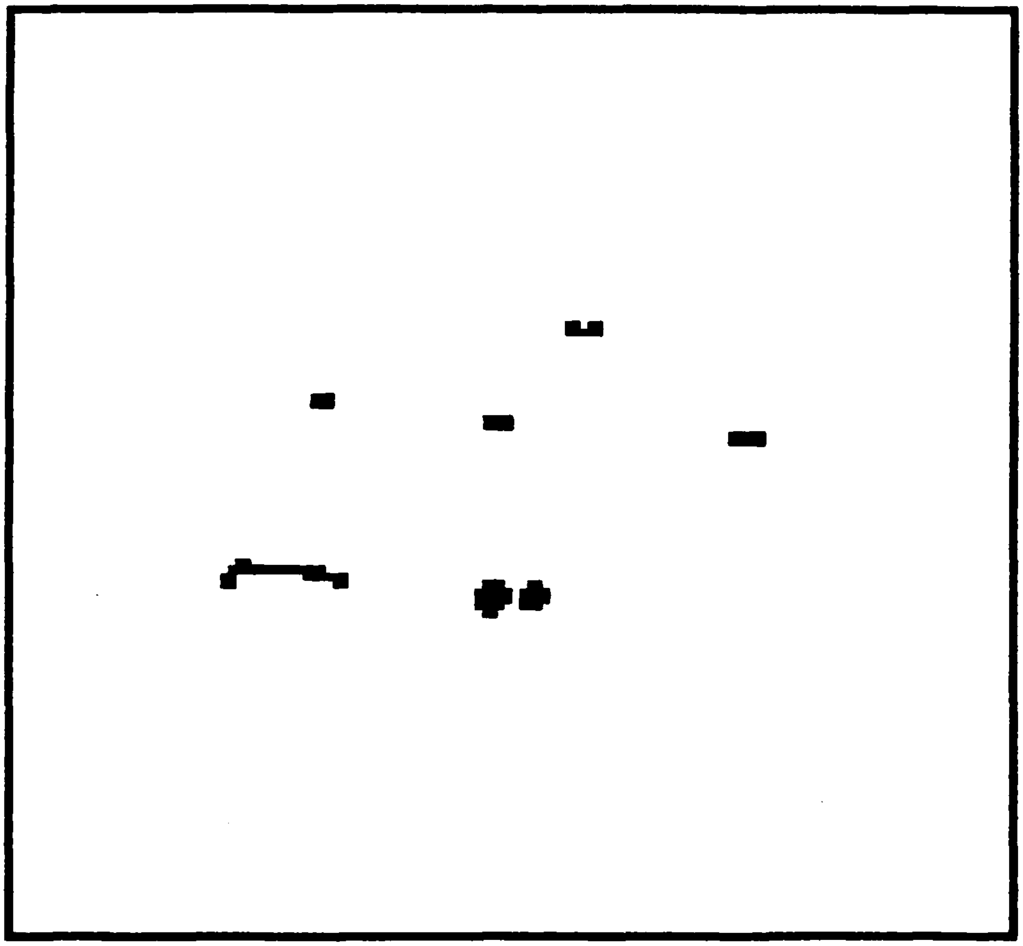


LOCALLY THRESHOLDED IMAG4.IR (7 X 7 NEIGHBORHOOD)

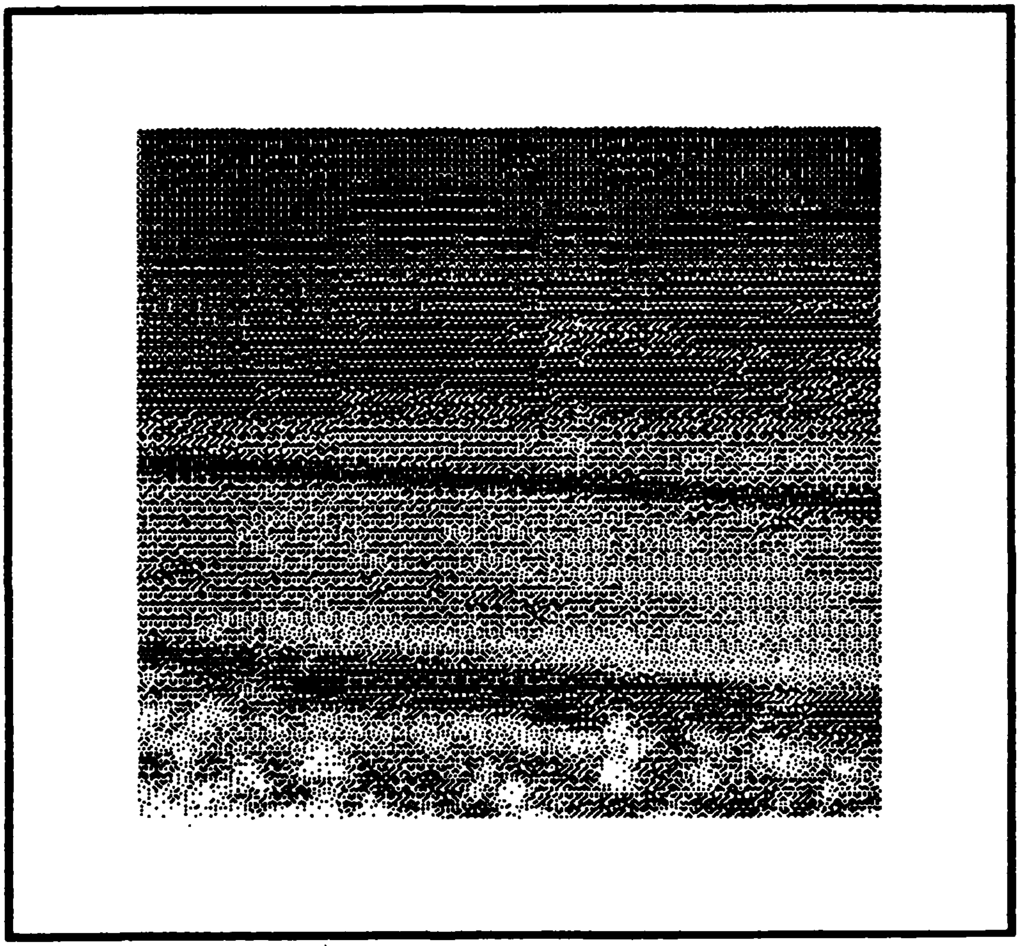


LOCALLY THRESHOLDED IMAG4.IR (7 X 7 NEIGHBORHOOD)
AFTER CONNECTIVITY TEST

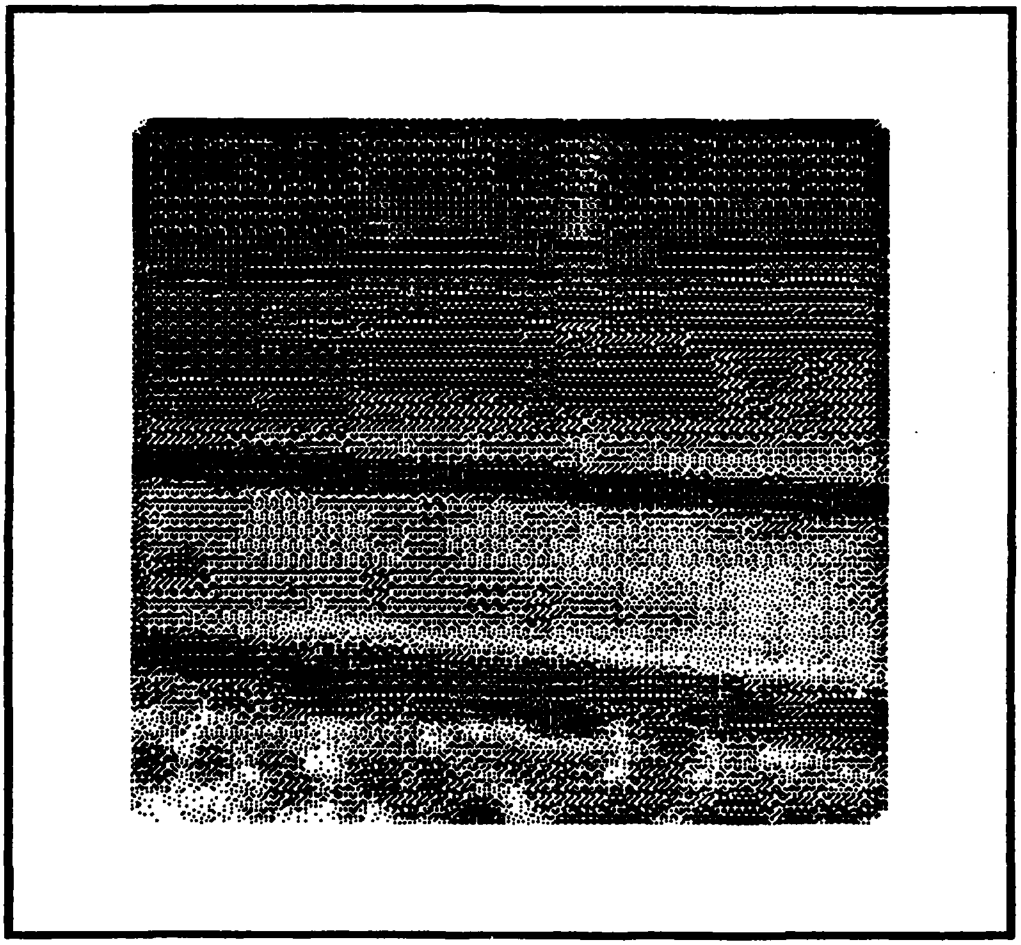
LOCALLY THRESHOLDED IMAG4.IR (USING 7 X 7 NEIGHBORHOOD WITHOUT EDGING)



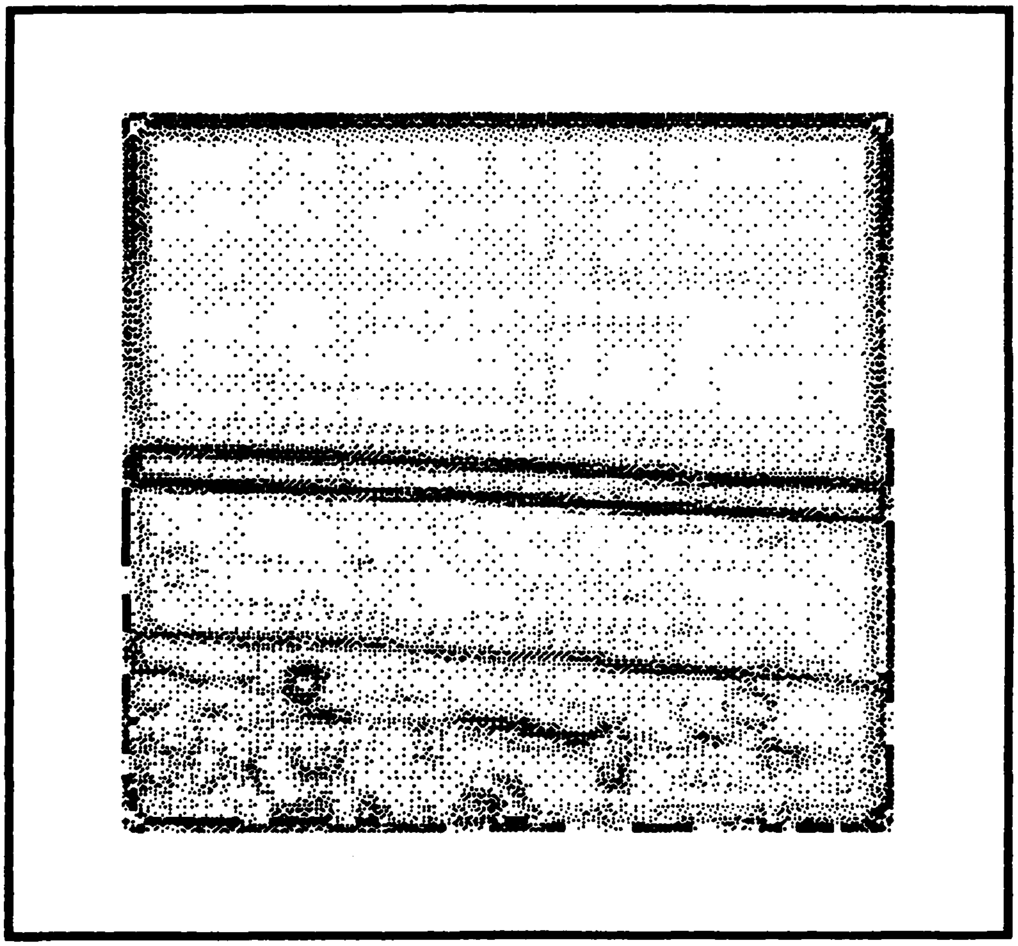
LOCALLY THRESHOLDED IMAG4.IR (USING 7 X 7 NEIGHBORHOOD WITHOUT EDGING) AFTER CONNECTIVITY TEST



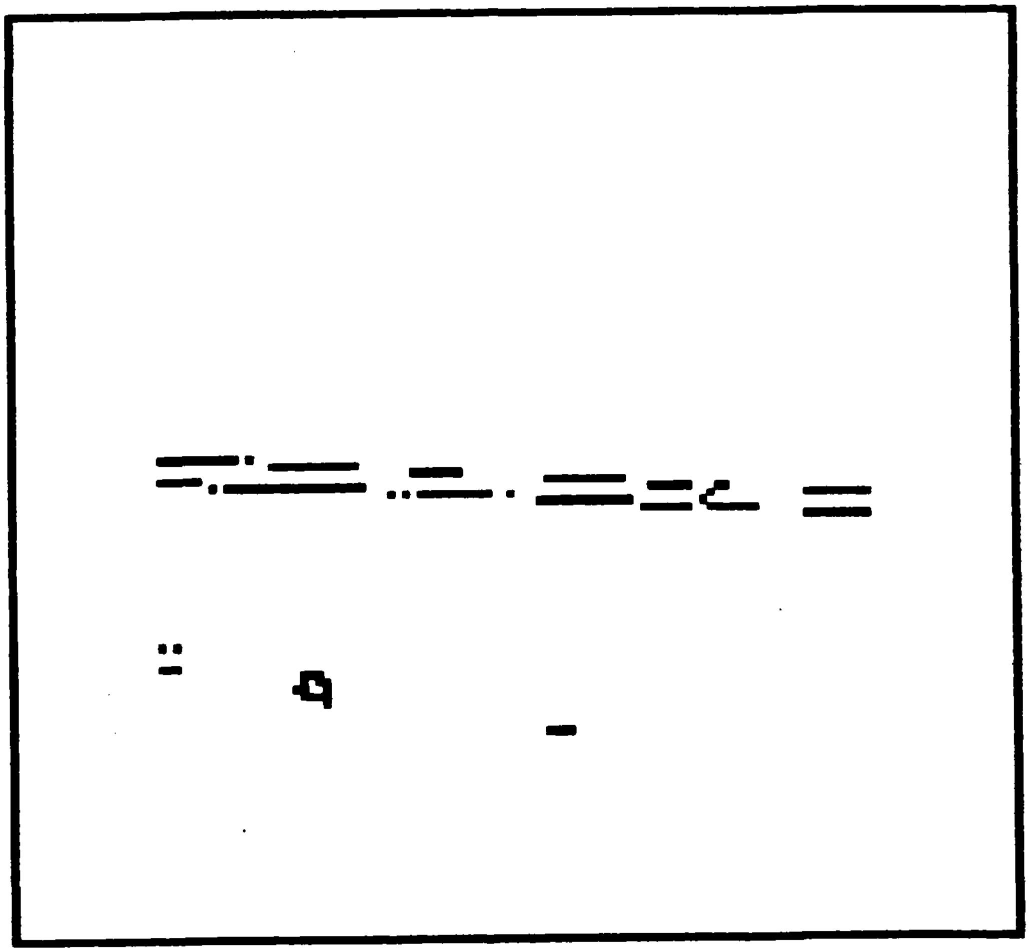
INFRARED IMAGE #5 (IMAG5. IR)



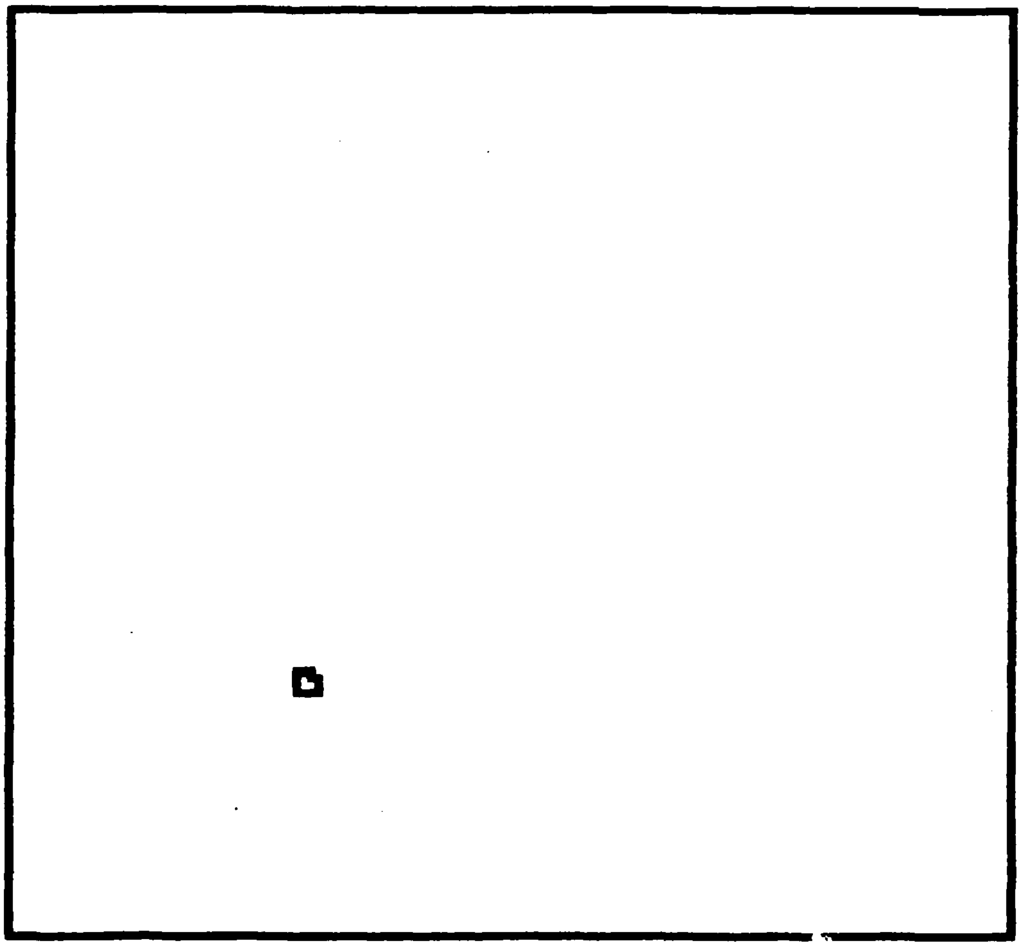
ENHANCED IMAGE OF IMAG5. IR



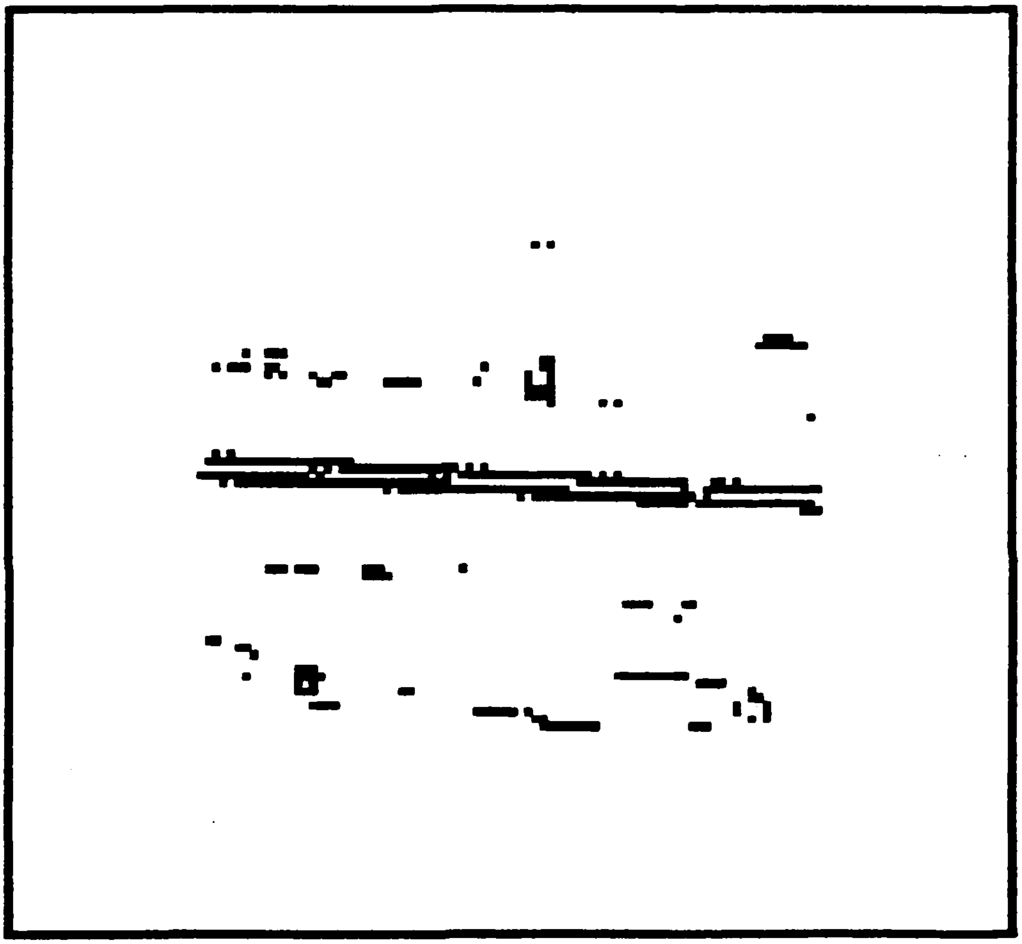
EDGED IMAGE OF IMAG5.IR



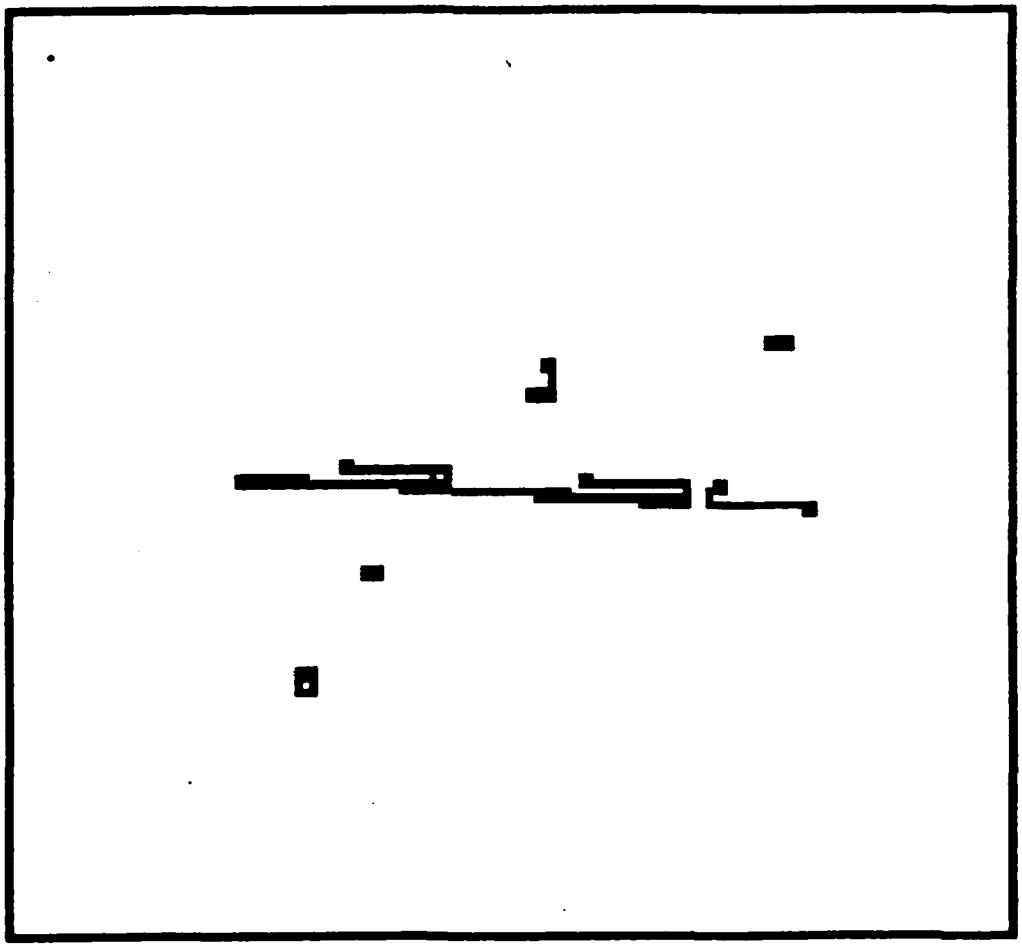
GLOBALLY THRESHOLDED IMAGE OF IMAG5.IR



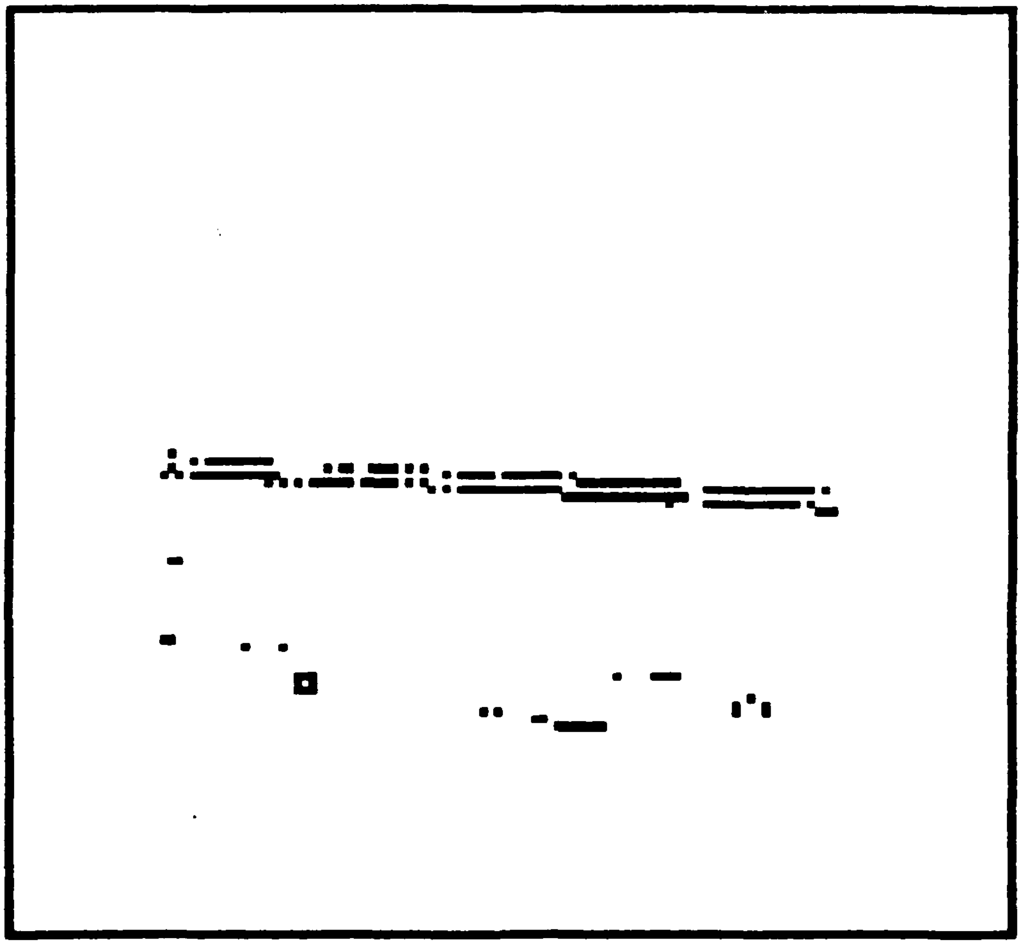
GLOBALLY THRESHOLDED IMAG5.IR AFTER CONNECTIVITY TEST



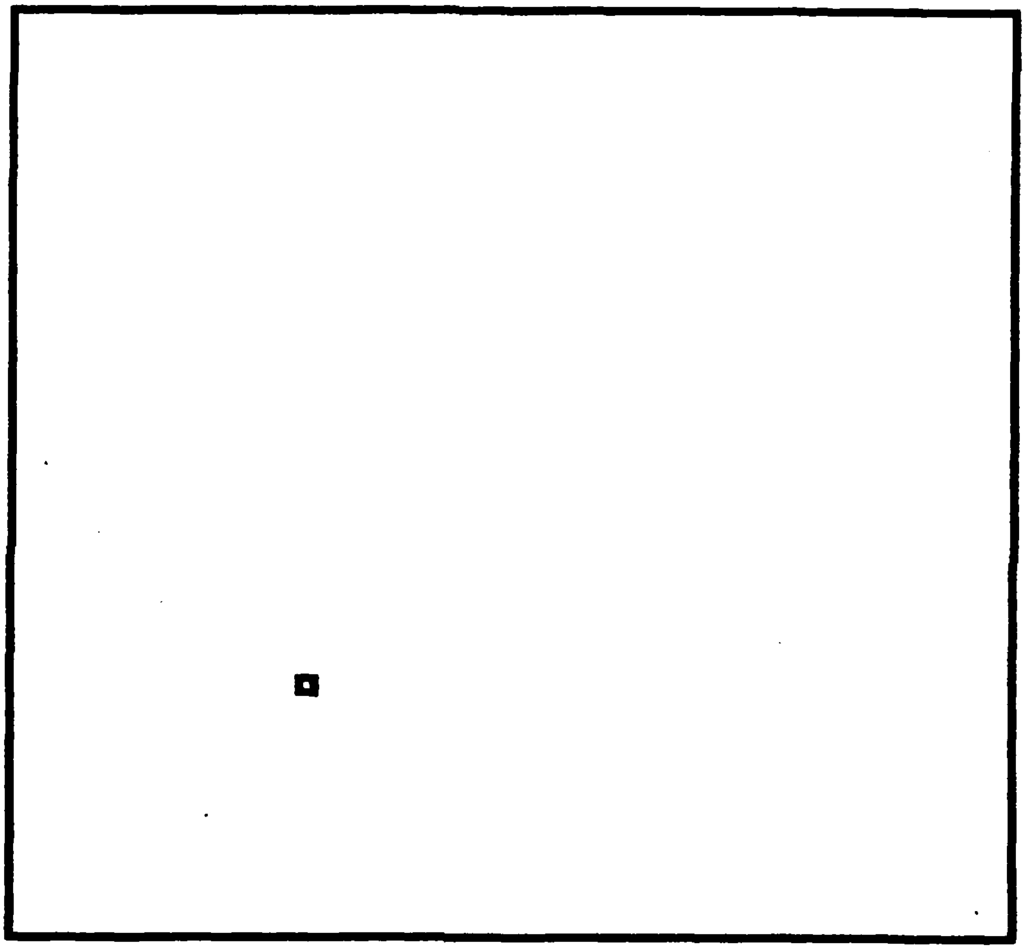
LOCALLY THRESHOLDED IMAG5.IR (17 X 17 NEIGHBORHOOD)



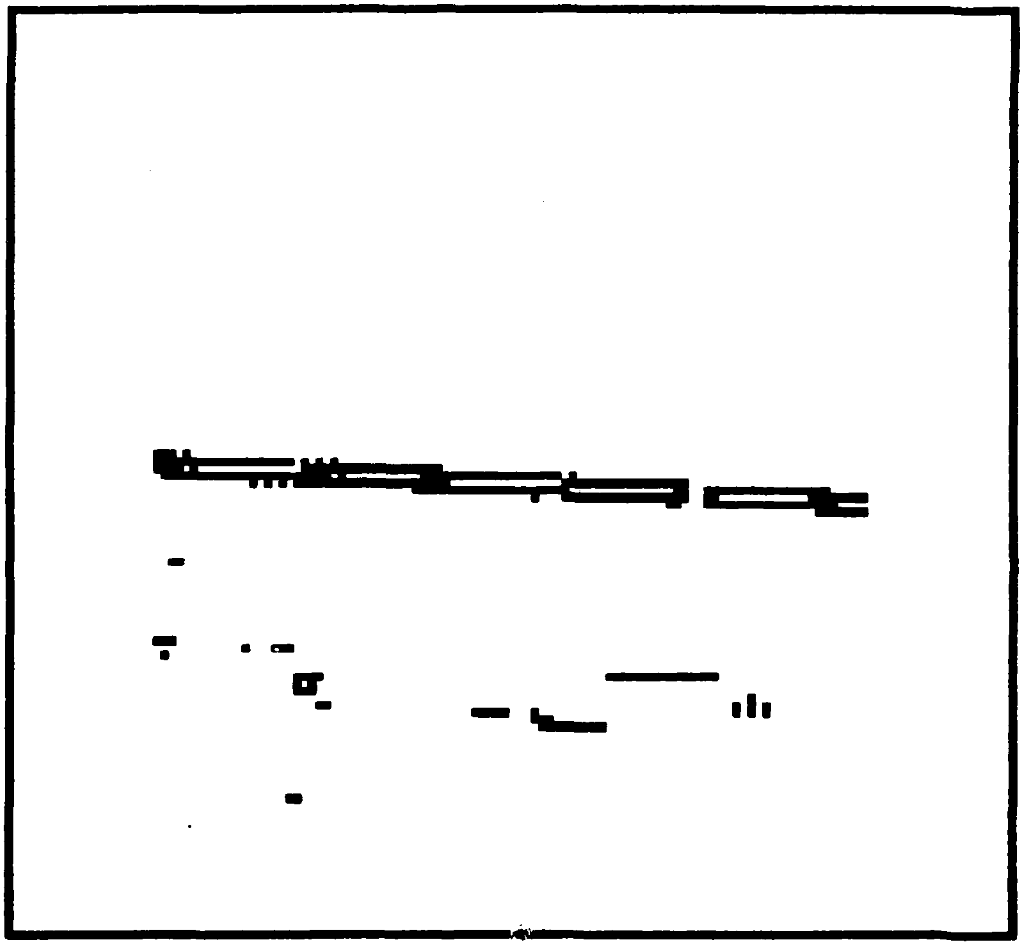
LOCALLY THRESHOLDED IMAG5.IR (17 X 17 NEIGHBORHOOD)
AFTER CONNECTIVITY TEST



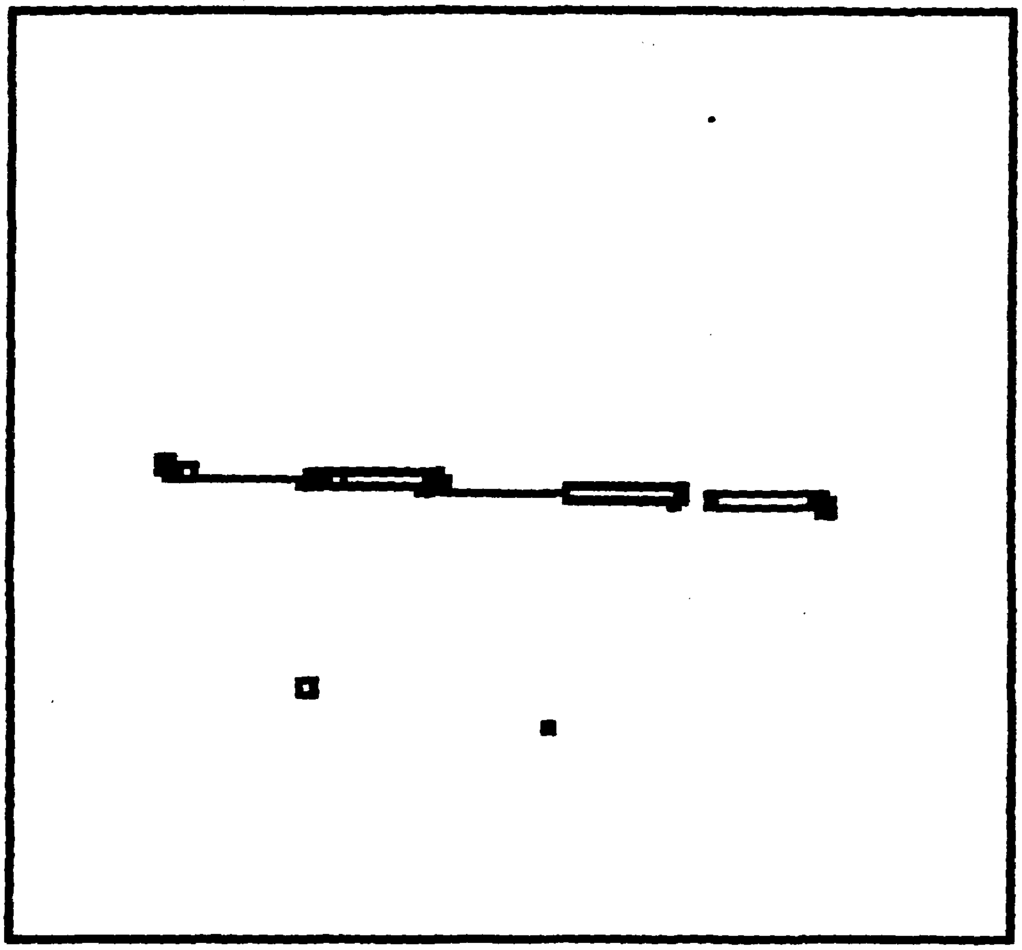
LOCALLY THRESHOLDED IMAG5.IR (7 X 7 NEIGHBORHOOD)



LOCALLY THRESHOLDED IMAGS.IR (7 X 7 NEIGHBORHOOD)
AFTER CONNECTIVITY TEST

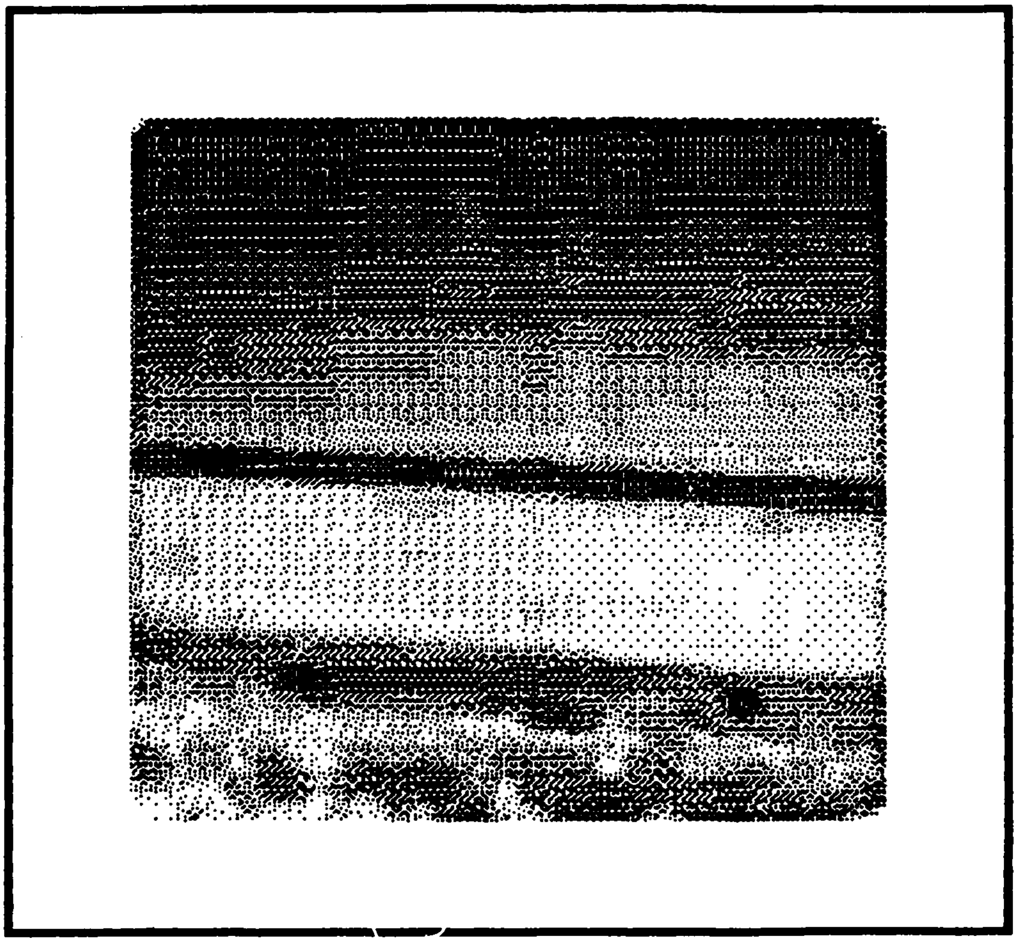


LOCALLY THRESHOLDED IMAG5.IR (USING 7 X 7 NEIGHBORHOOD
WITHOUT EDGING)

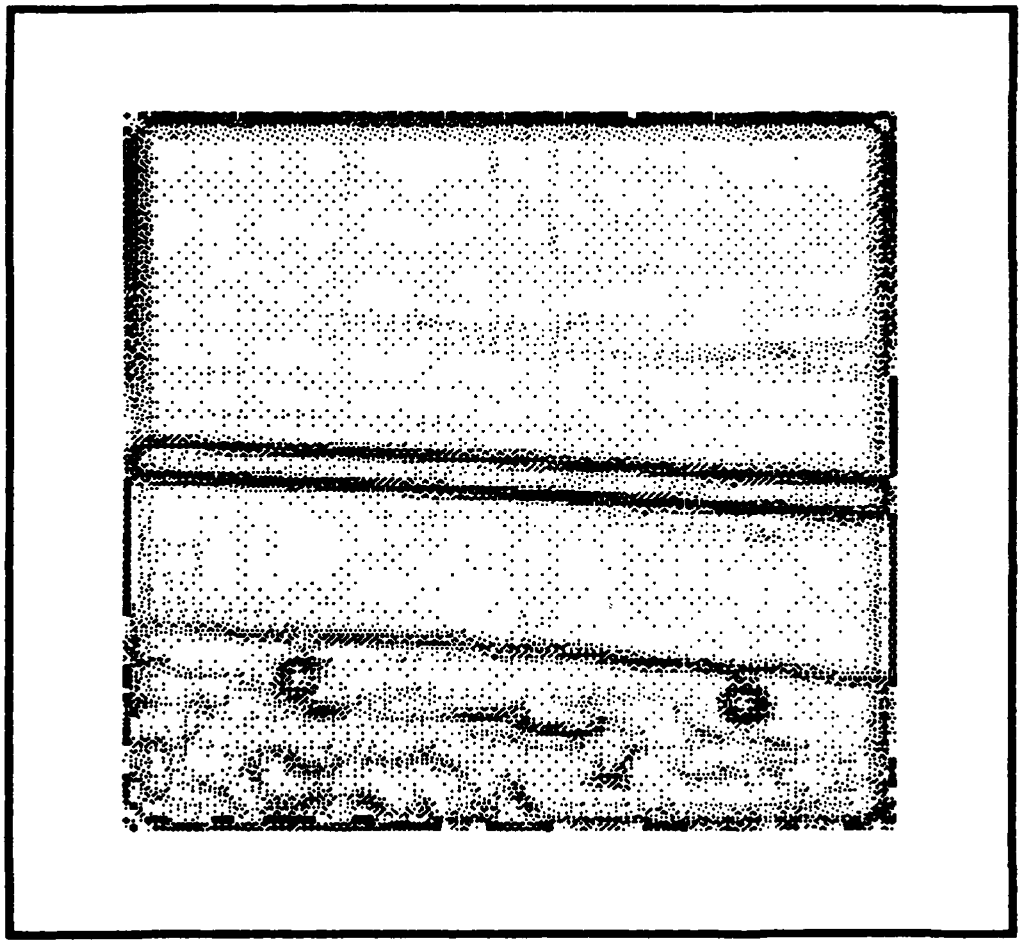


LOCALLY THRESHOLDED IMAG5.IR (USING 7 X 7 NEIGHBORHOOD
WITHOUT EDGING) AFTER CONNECTIVITY TEST

INFRARED IMAGE #6 (IMAG6.IR)



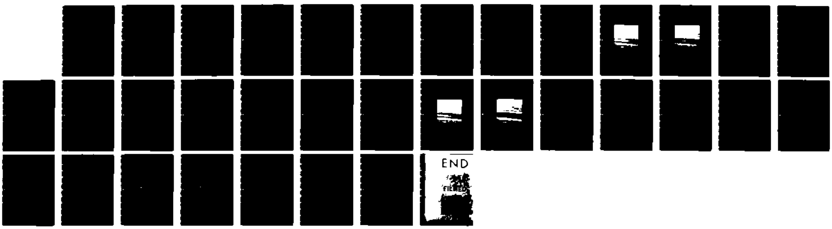
ENHANCED IMAGE OF IMAG6. IR

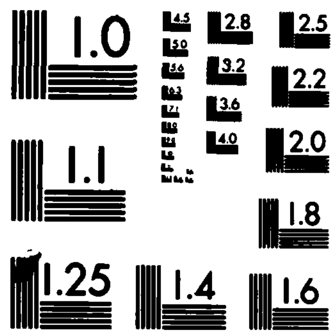


EDGED IMAGE OF IMAG6.IR

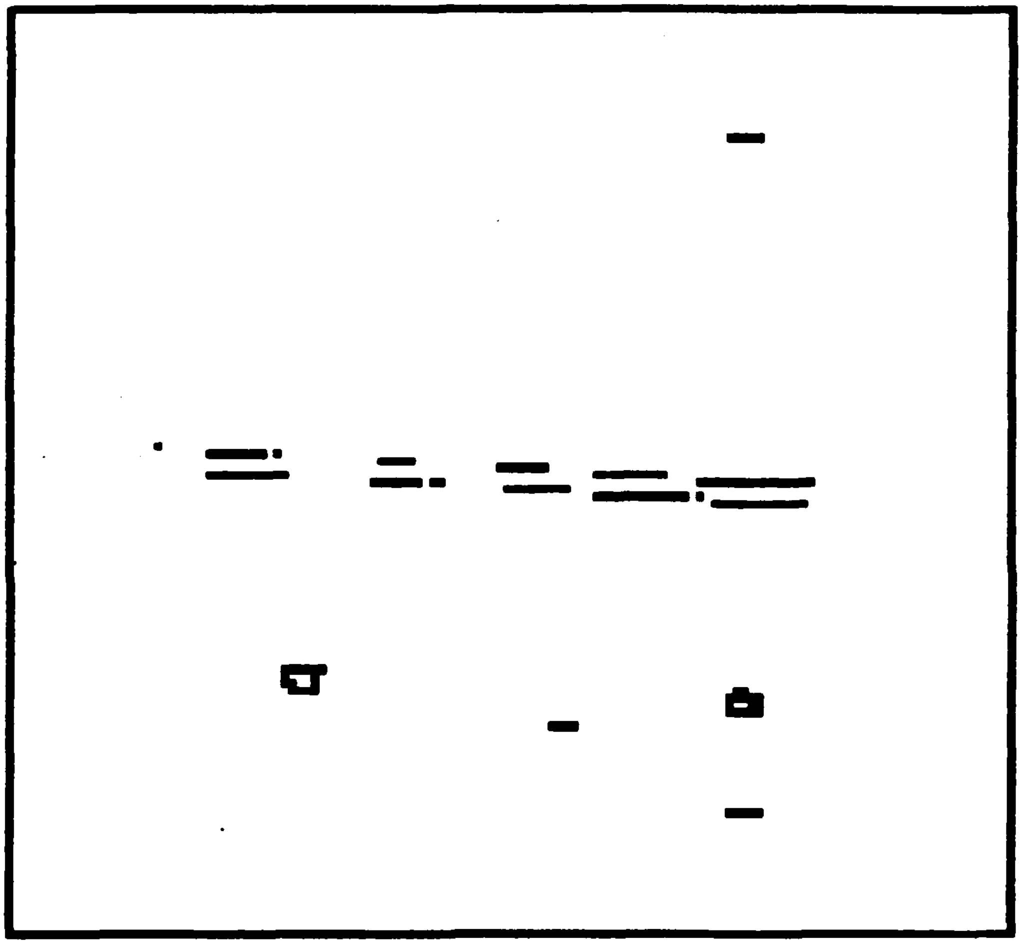
AD-A138 421 IMAGE FILTERING WITH BOOLEAN AND STATISTICAL OPERATORS 3/3
(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH
SCHOOL OF ENGINEERING R D WELLS DEC 83

UNCLASSIFIED AFIT/GE/EE/83D-72 F/G 20/6 NL

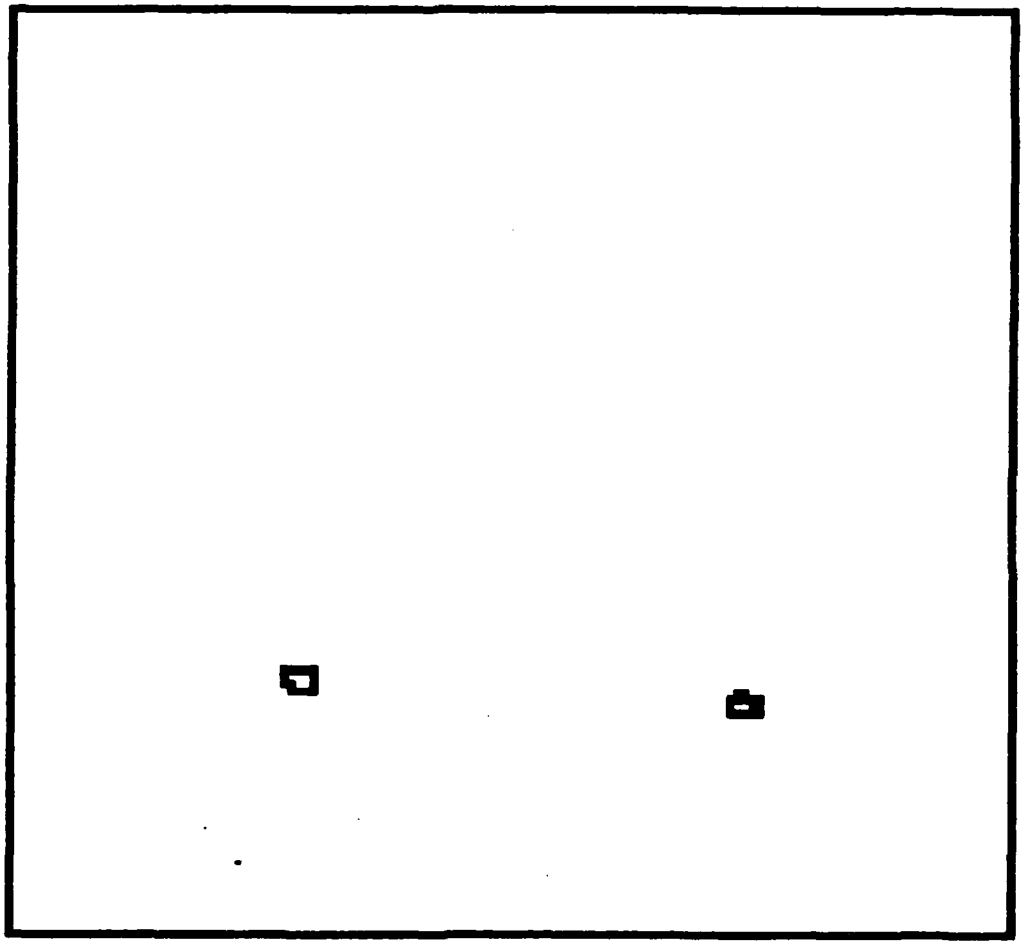




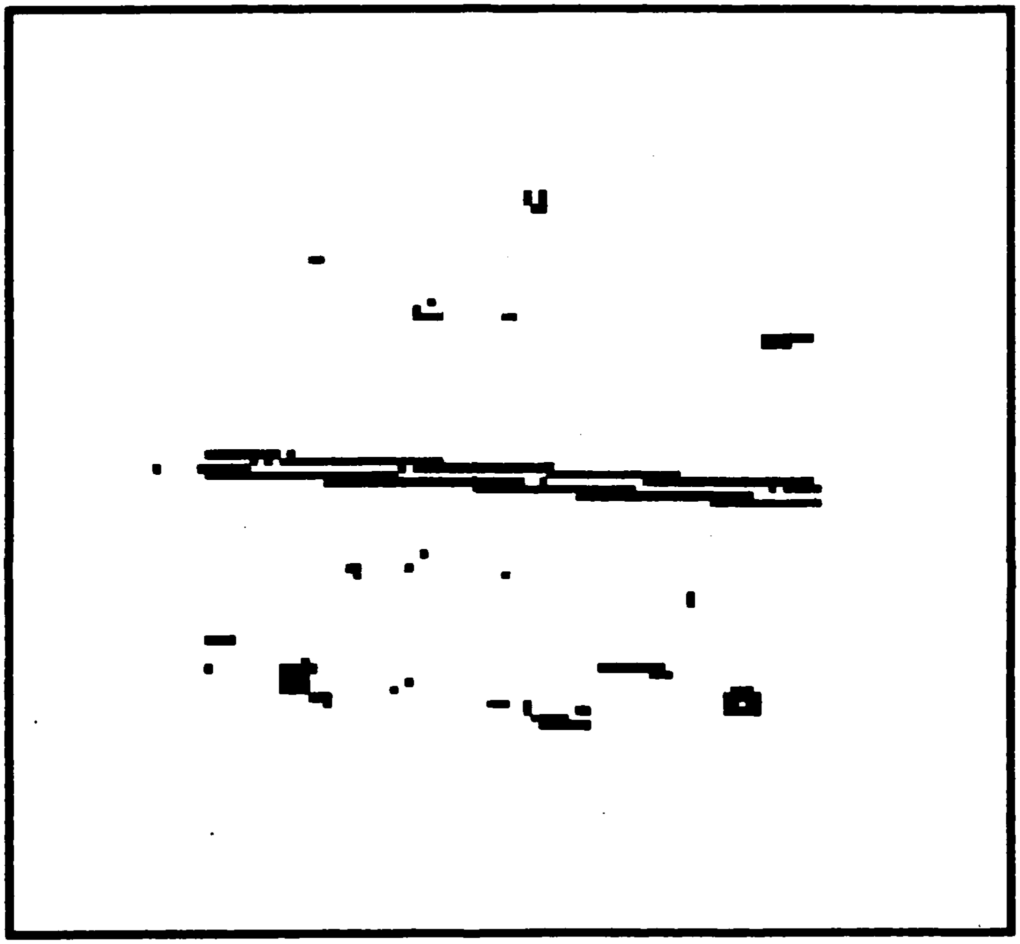
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



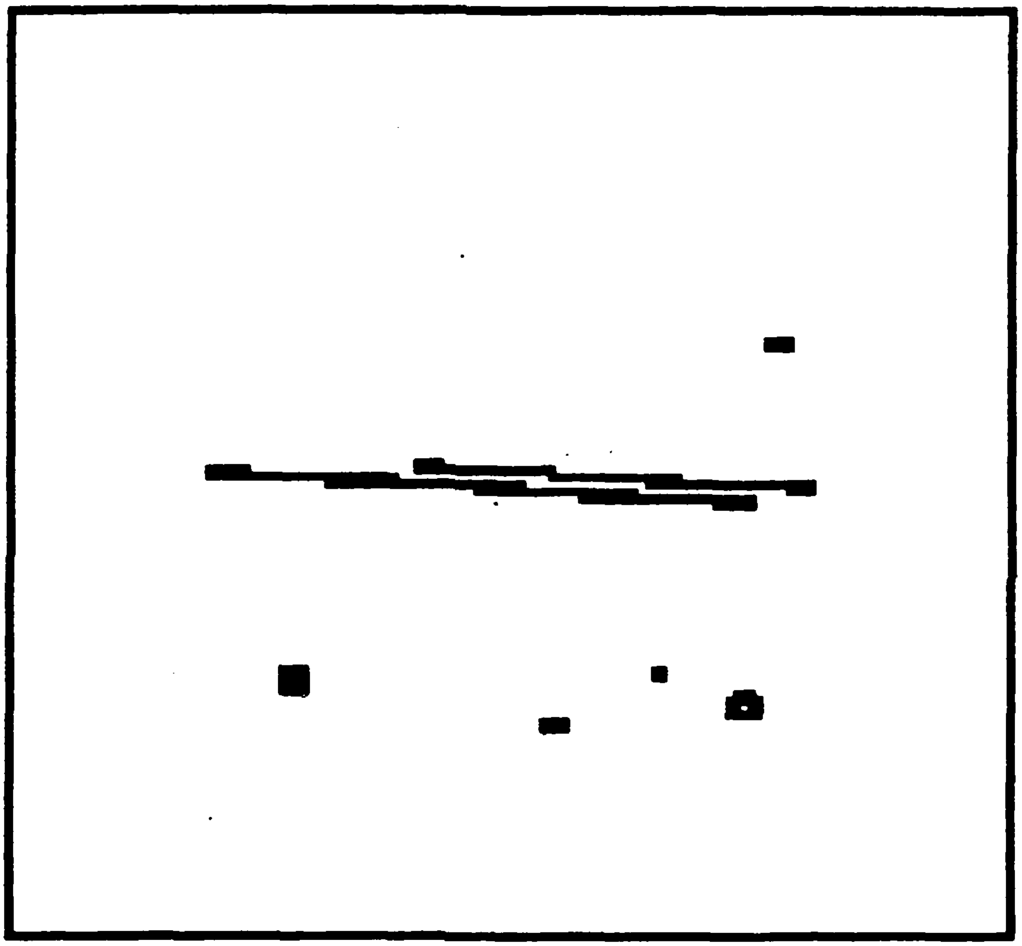
GLOBALLY THRESHOLDED IMAGE OF IMAG6.IR



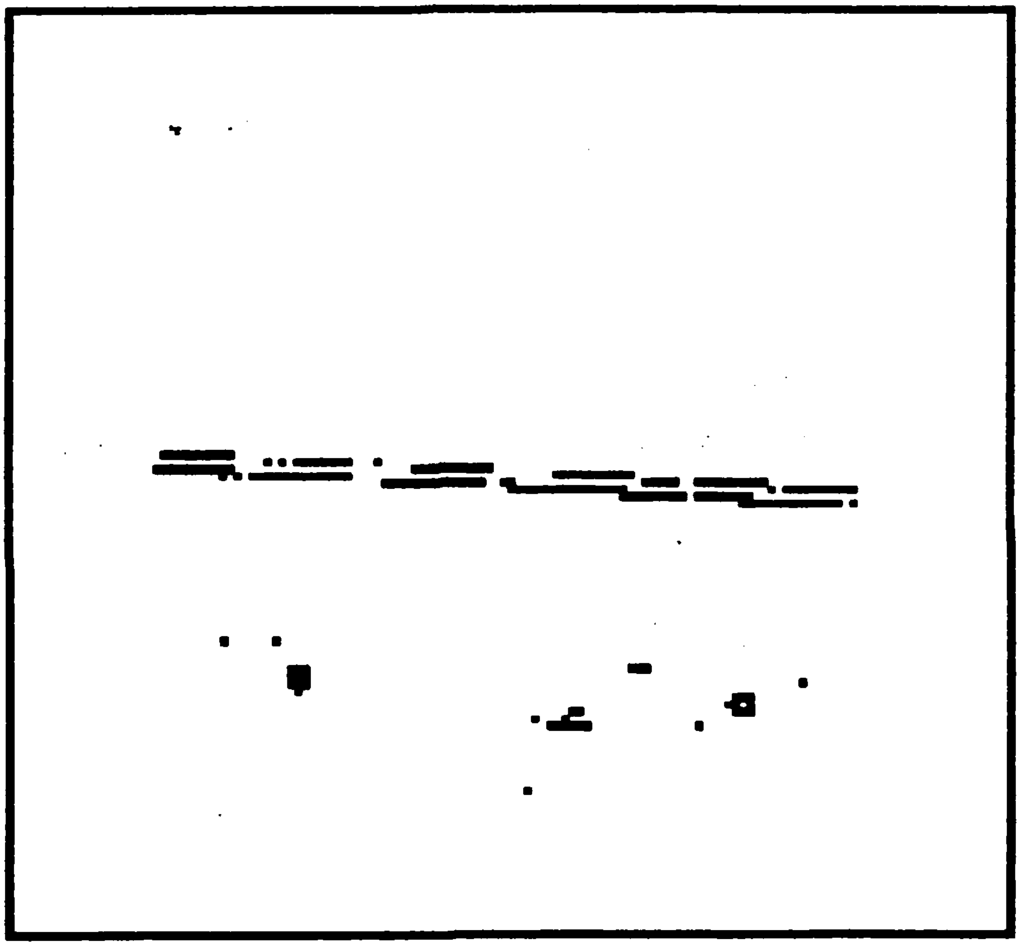
GLOBALLY THRESHOLDED IMAG6.IR AFTER CONNECTIVITY TEST



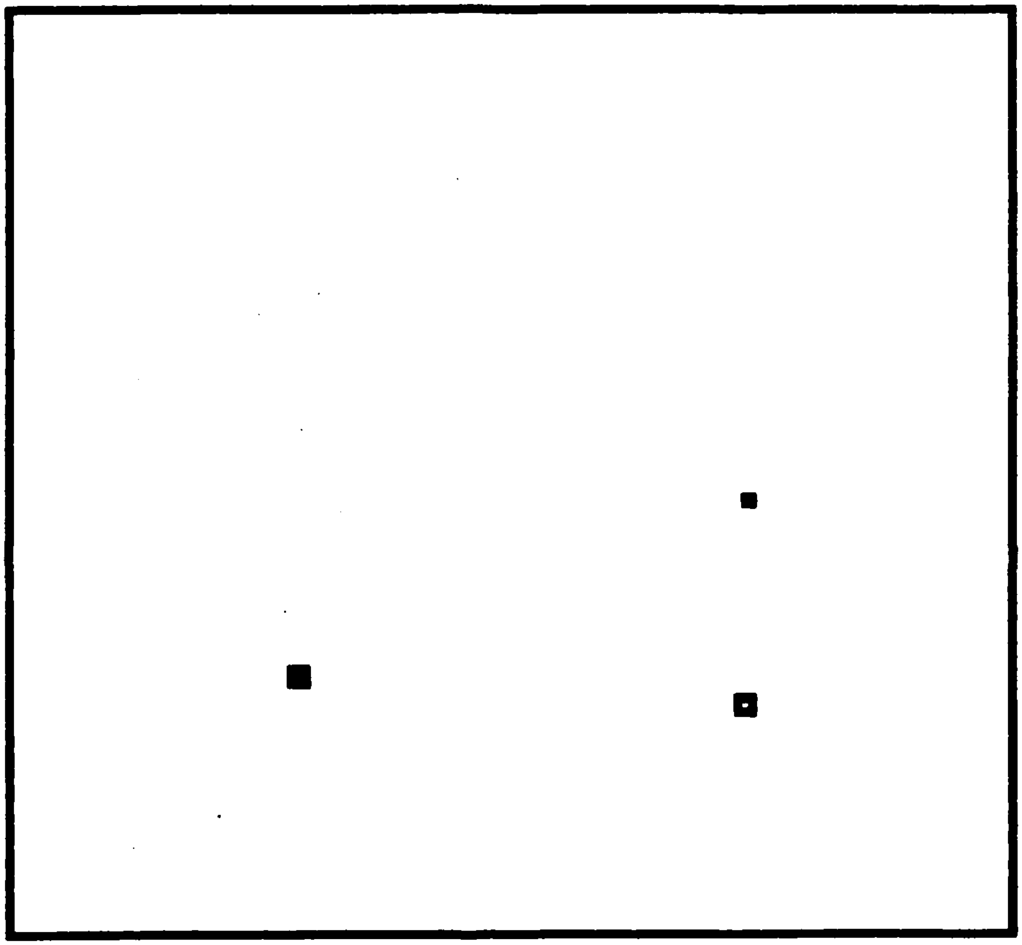
LOCALLY THRESHOLDED IMAG6.IR (17 X 17 NEIGHBORHOOD)



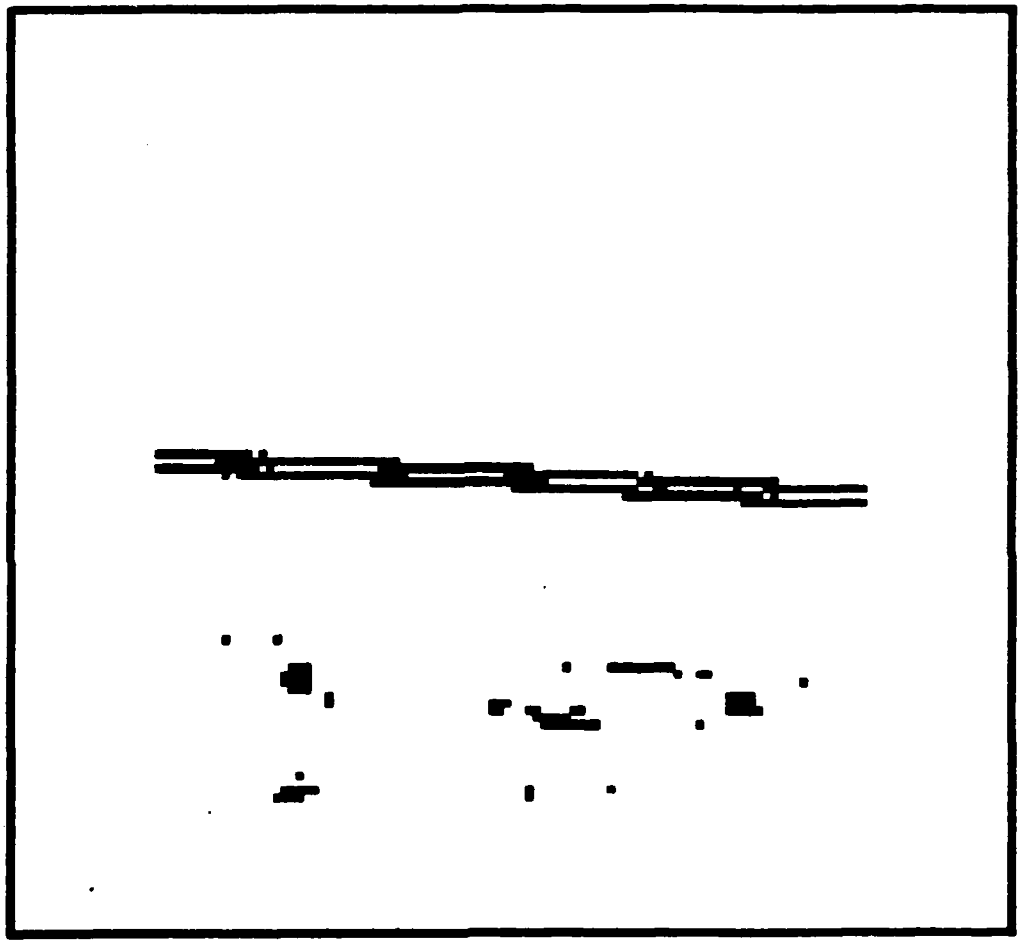
LOCALLY THRESHOLDED IMAG6.IR (17 X 17 NEIGHBORHOOD)
AFTER CONNECTIVITY TEST



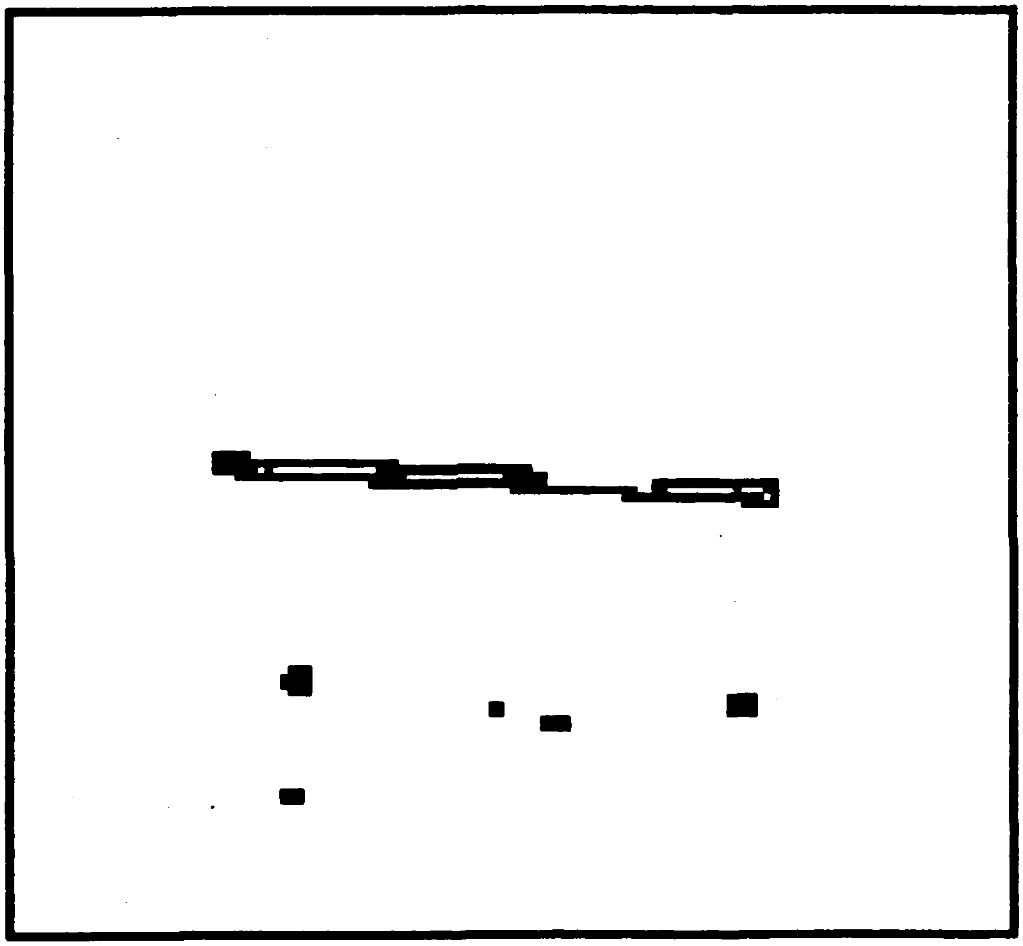
LOCALLY THRESHOLDED IMAG6.IR (7 X 7 NEIGHBORHOOD)



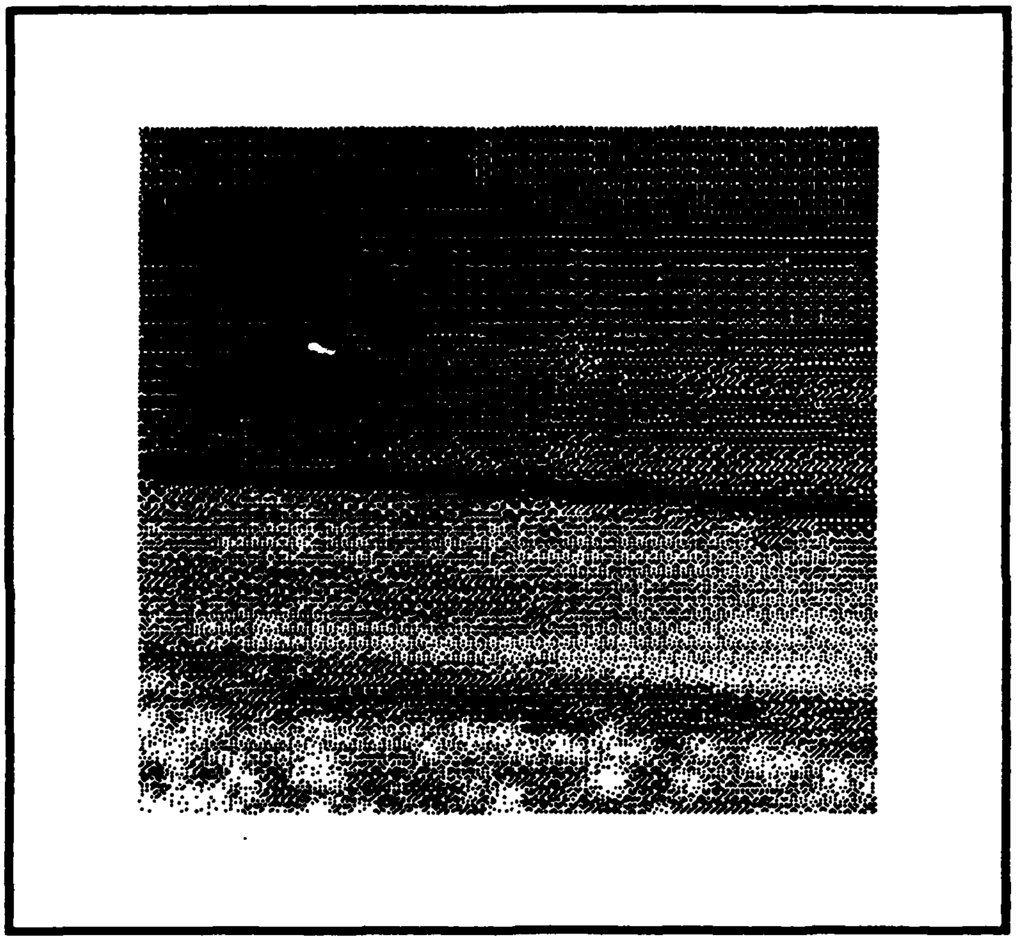
LOCALLY THRESHOLDED IMAG6.IR (7 X 7 NEIGHBORHOOD)
AFTER CONNECTIVITY TEST



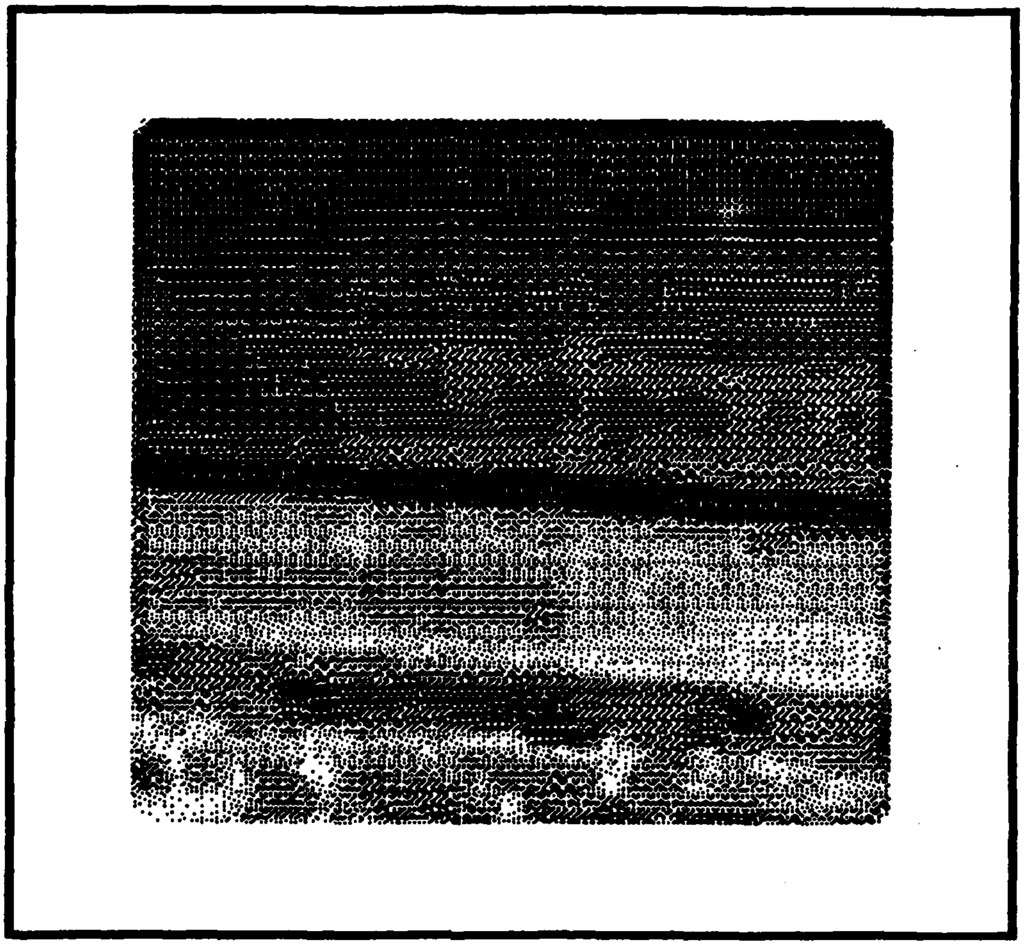
LOCALLY THRESHOLDED IMAG6. IR (USING 7 X 7 NEIGHBORHOOD
WITHOUT EDGING)



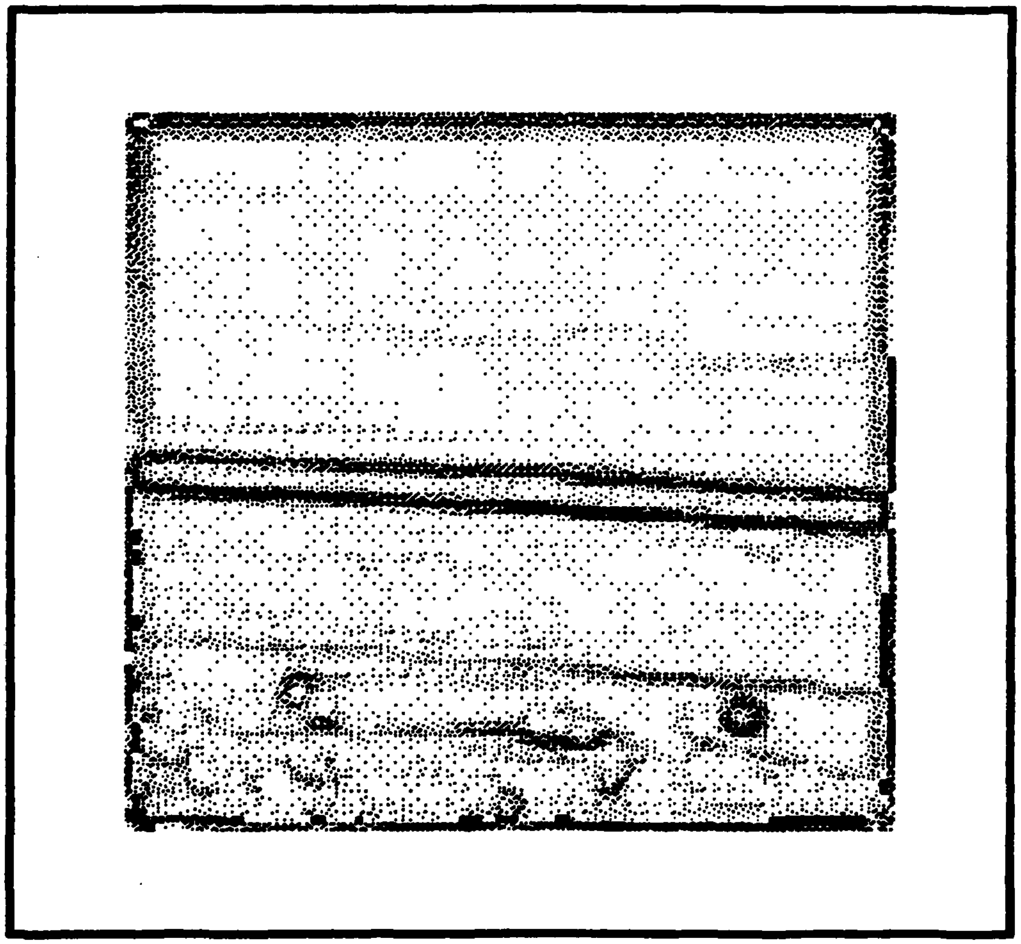
LOCALLY THRESHOLDED IMAG6. IR (USING 7 X 7 NEIGHBORHOOD WITHOUT EDGING) AFTER CONNECTIVITY TEST



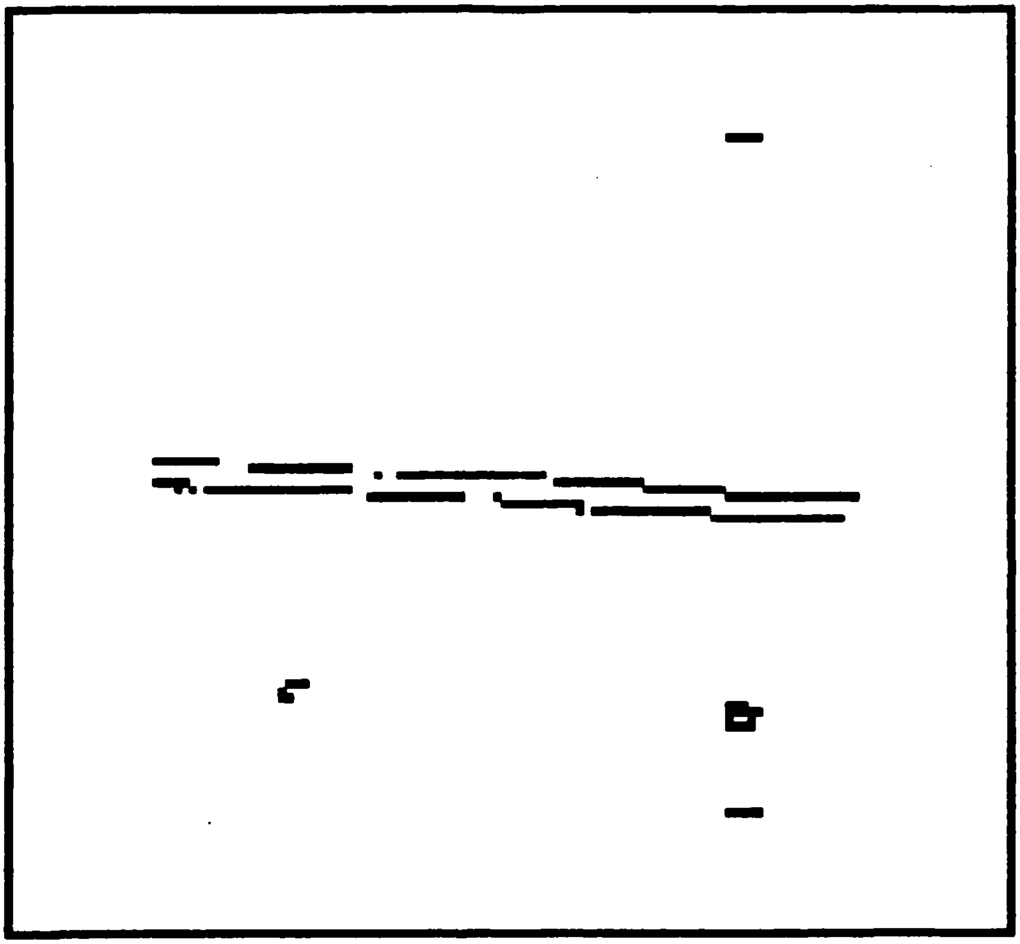
INFRARED IMAGE #7 (IMAG7.IR)



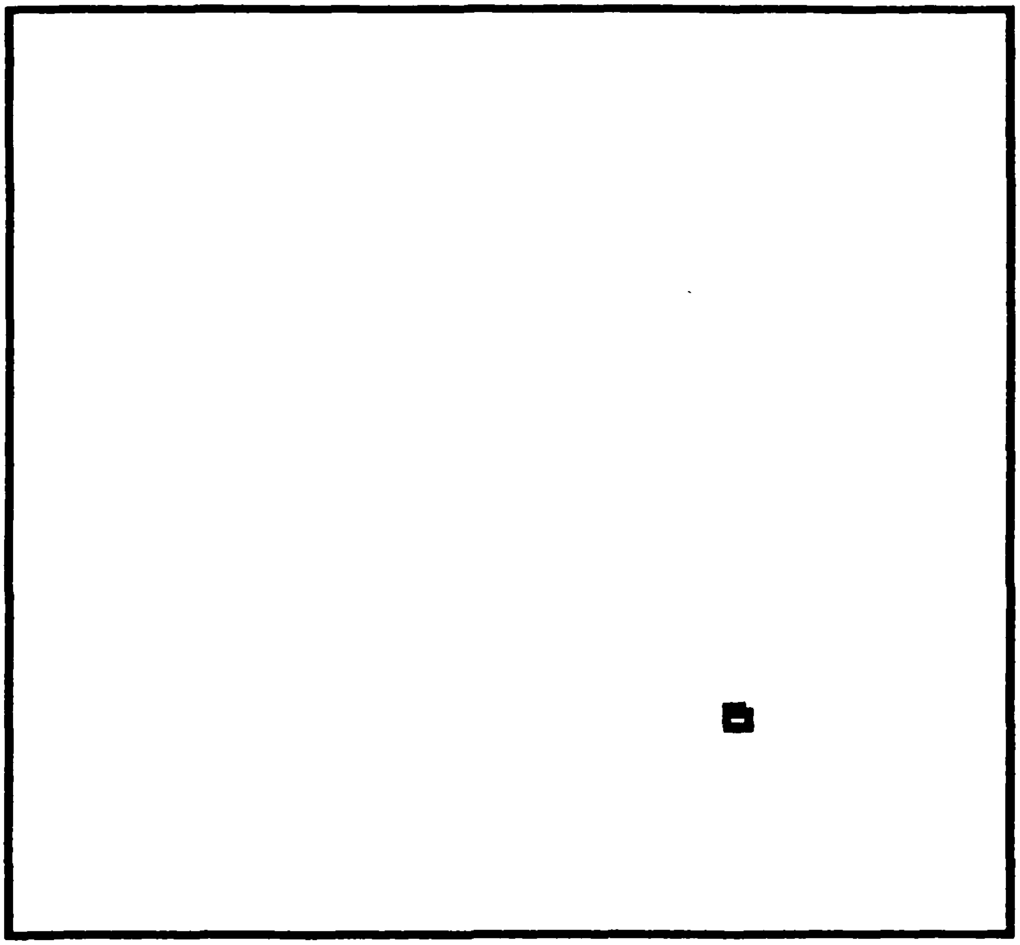
ENHANCED IMAGE OF IMAG7.IR



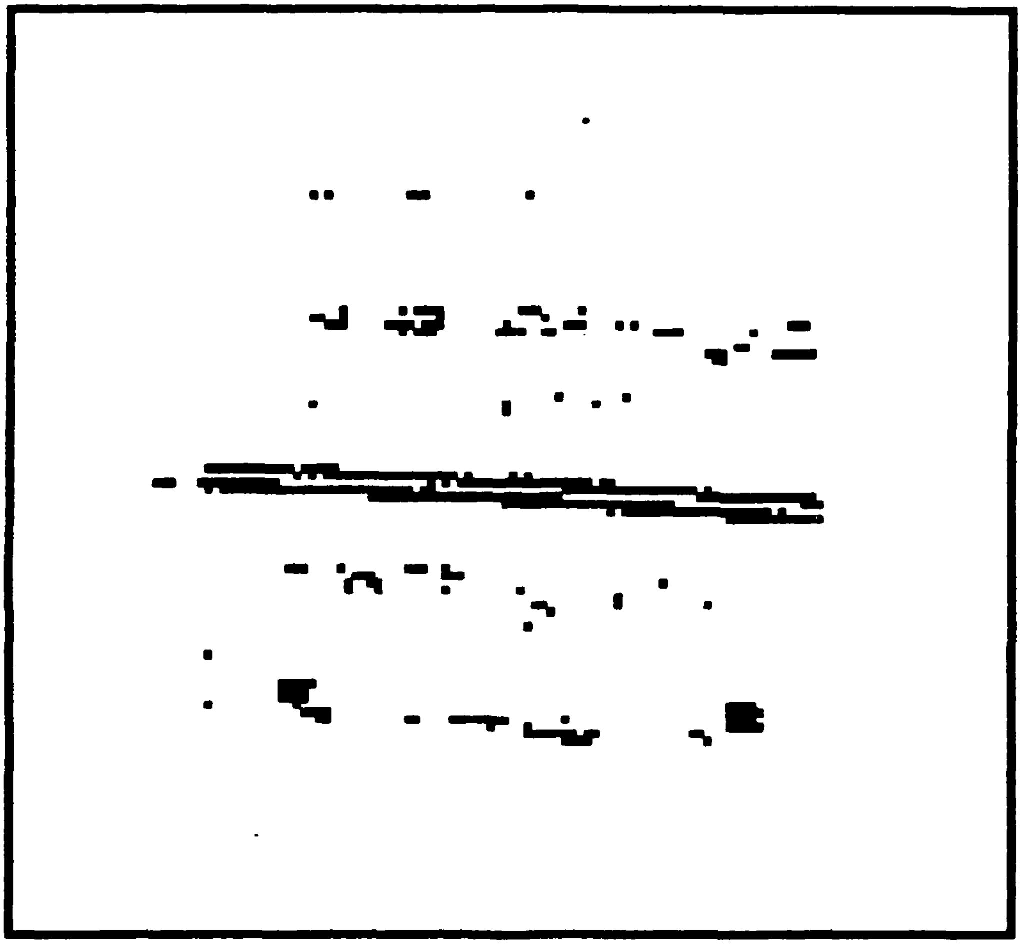
EDGED IMAGE OF IMAG7. IR



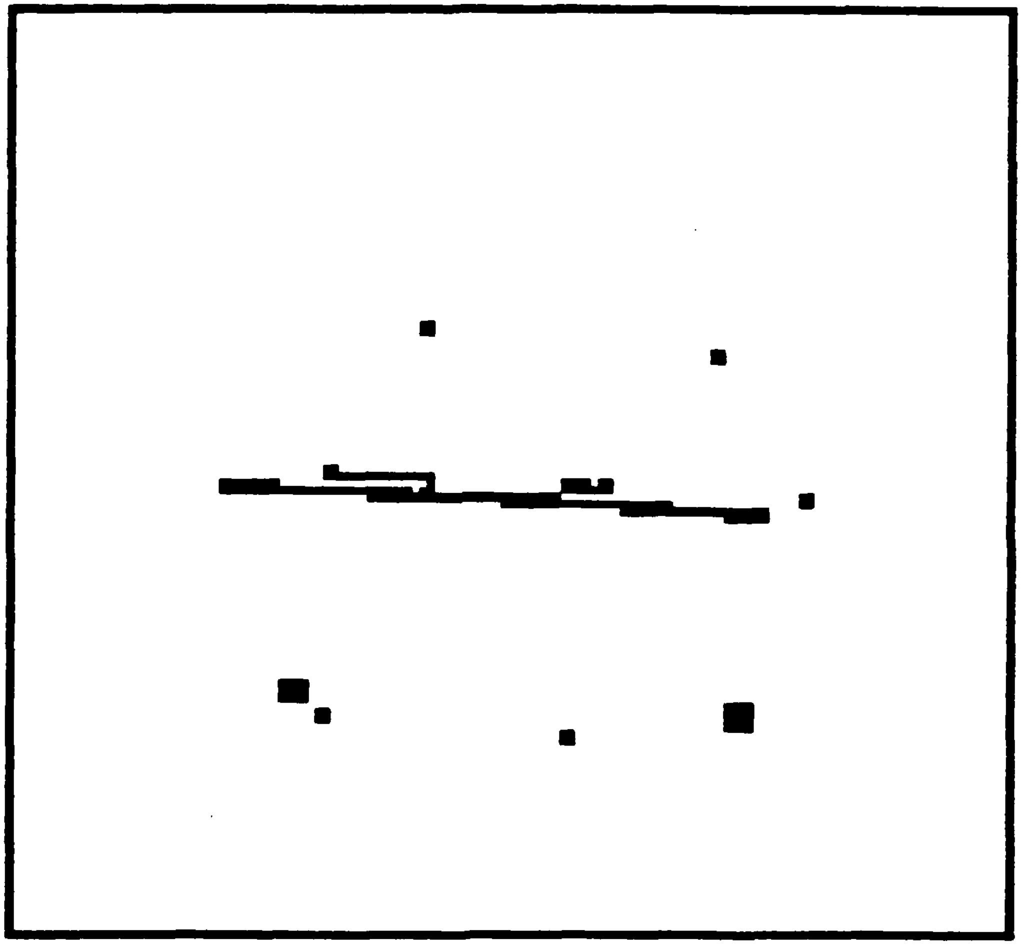
GLOBALLY THRESHOLDED IMAGE OF IMAG7.IR



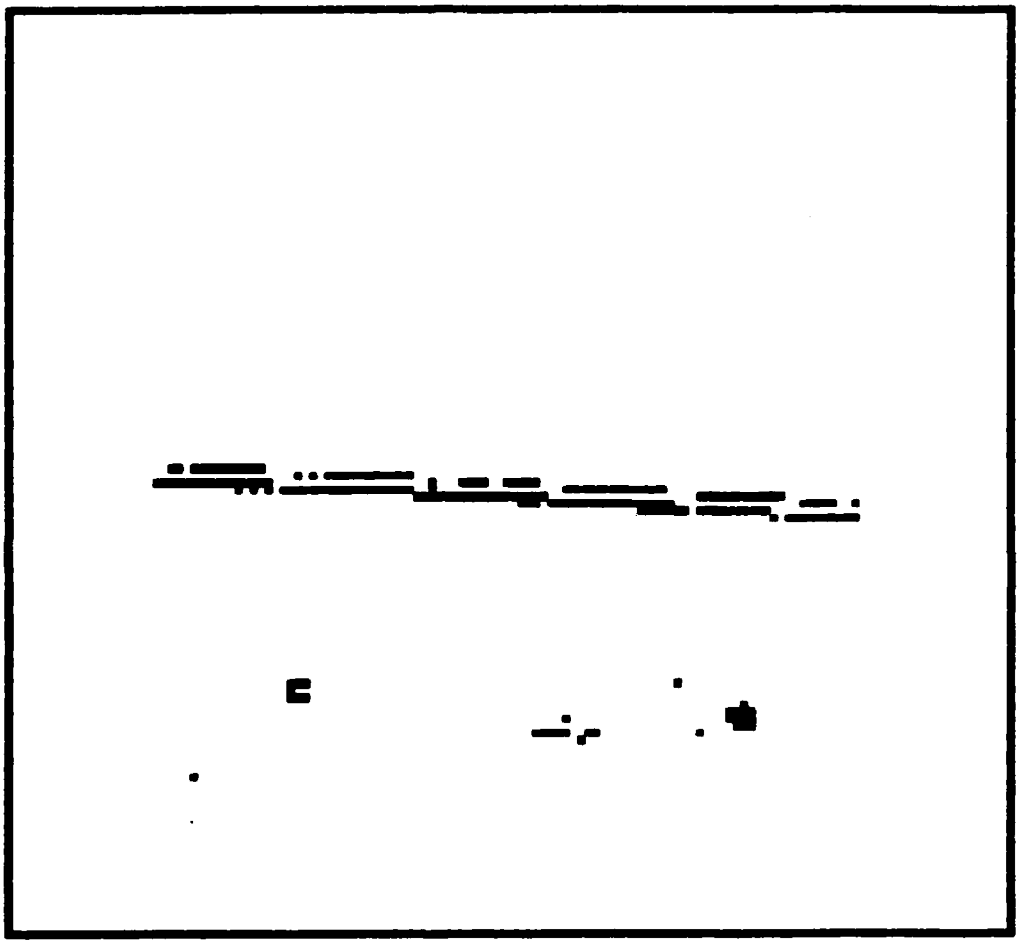
GLOBALLY THRESHOLDED IMAG7.IR AFTER CONNECTIVITY TEST



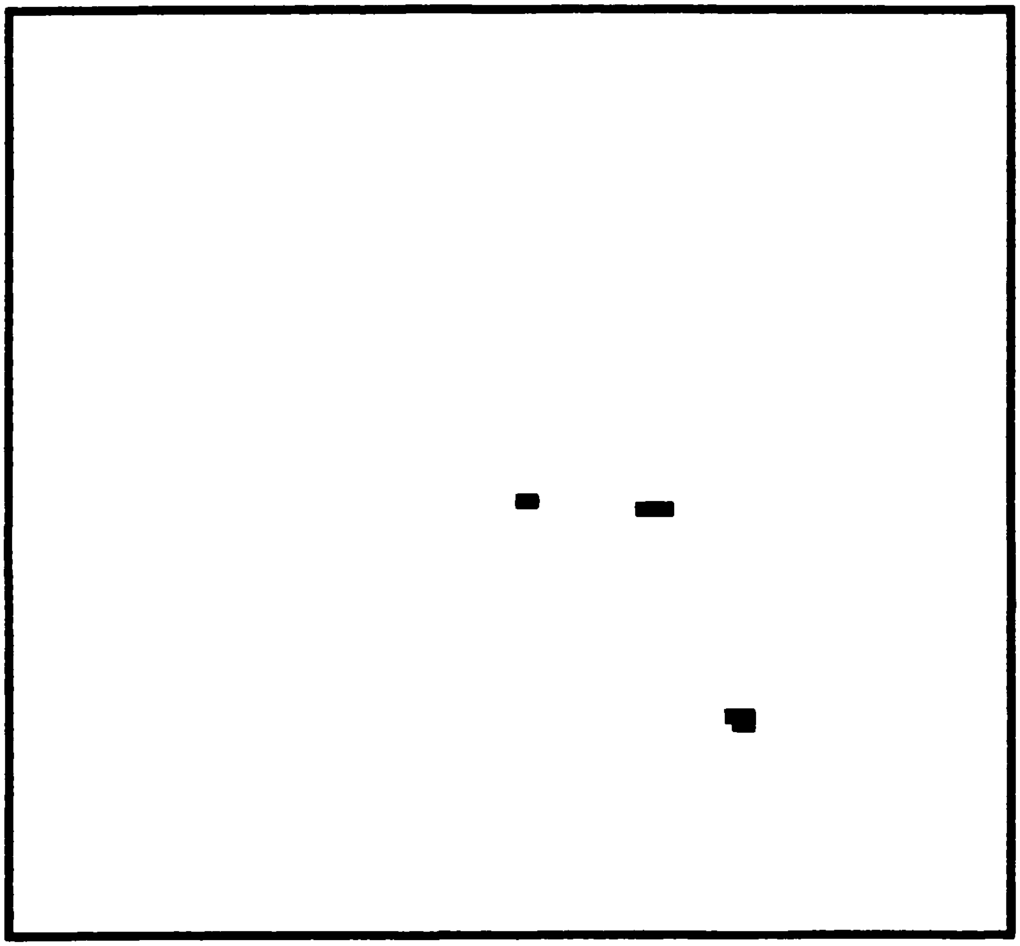
LOCALLY THRESHOLDED IMAG7.IR (17 X 17 NEIGHBORHOOD)



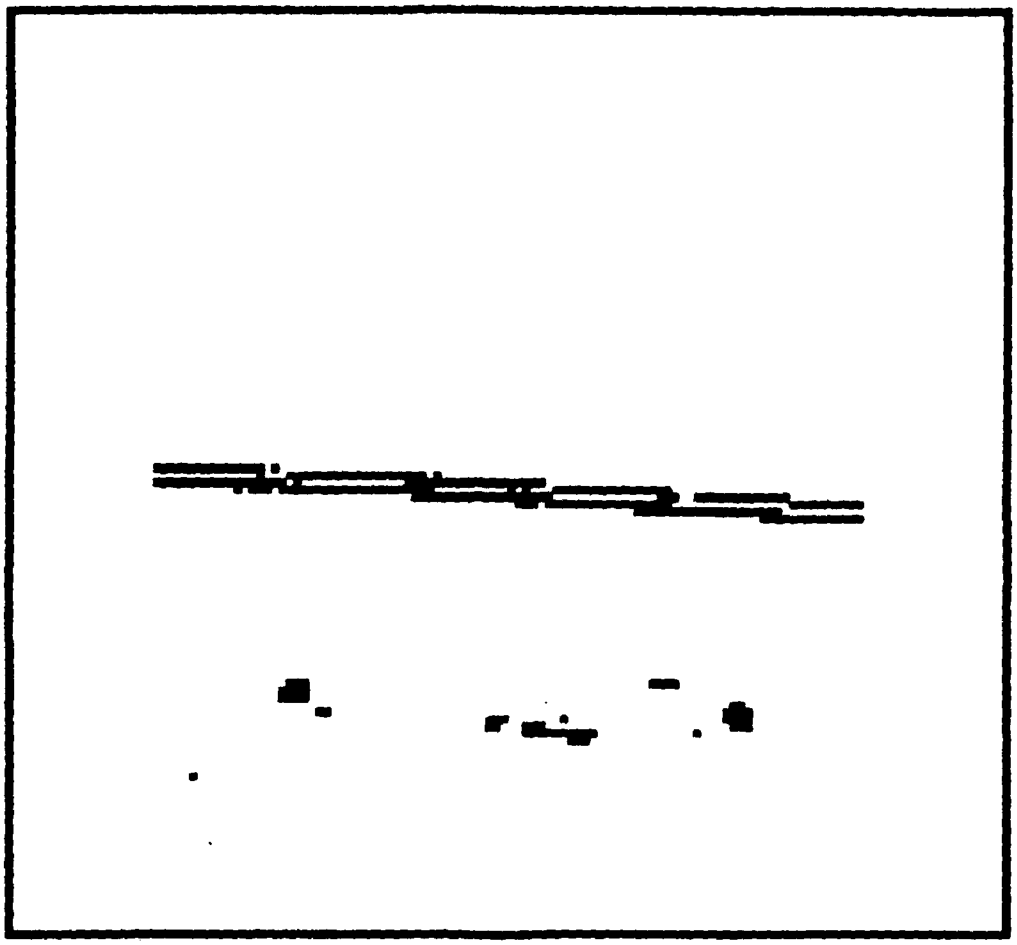
LOCALLY THRESHOLDED IMA07.IR (17 X 17 NEIGHBORHOOD)
AFTER CONNECTIVITY TEST



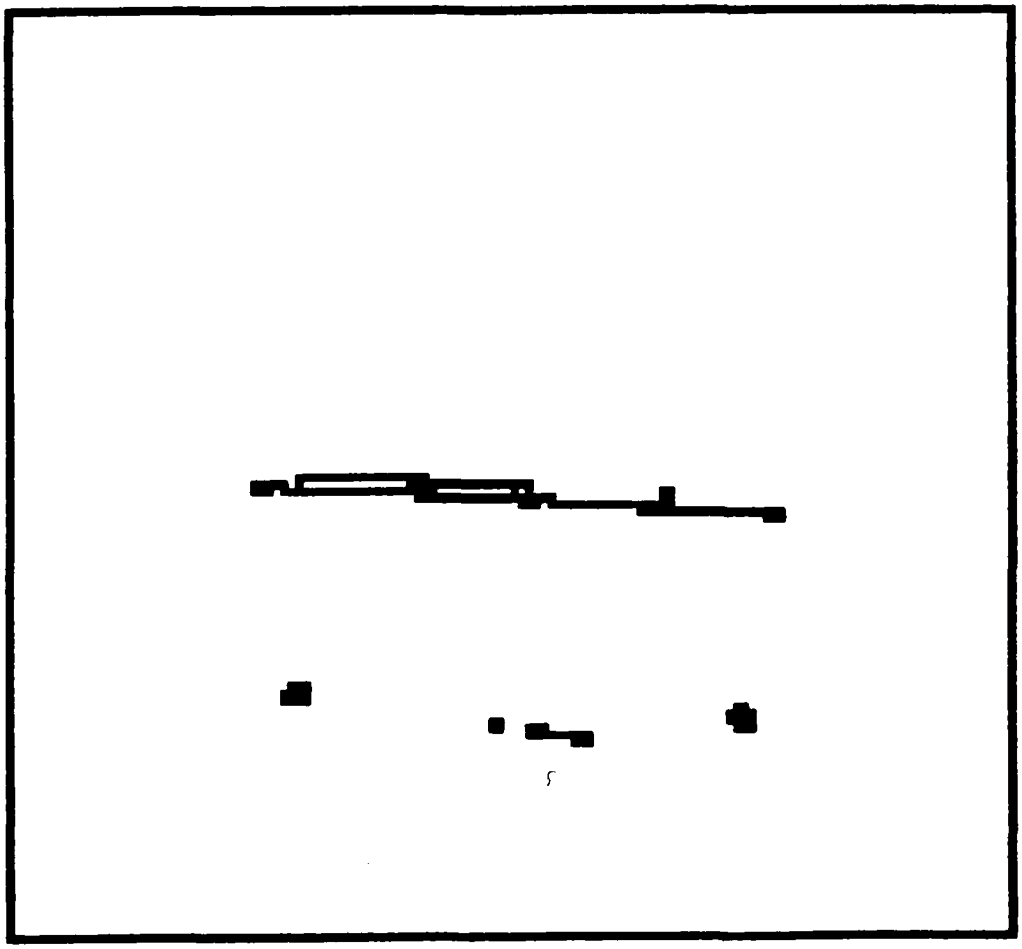
LOCALLY THRESHOLDED IMAG7.IR (7 X 7 NEIGHBORHOOD)



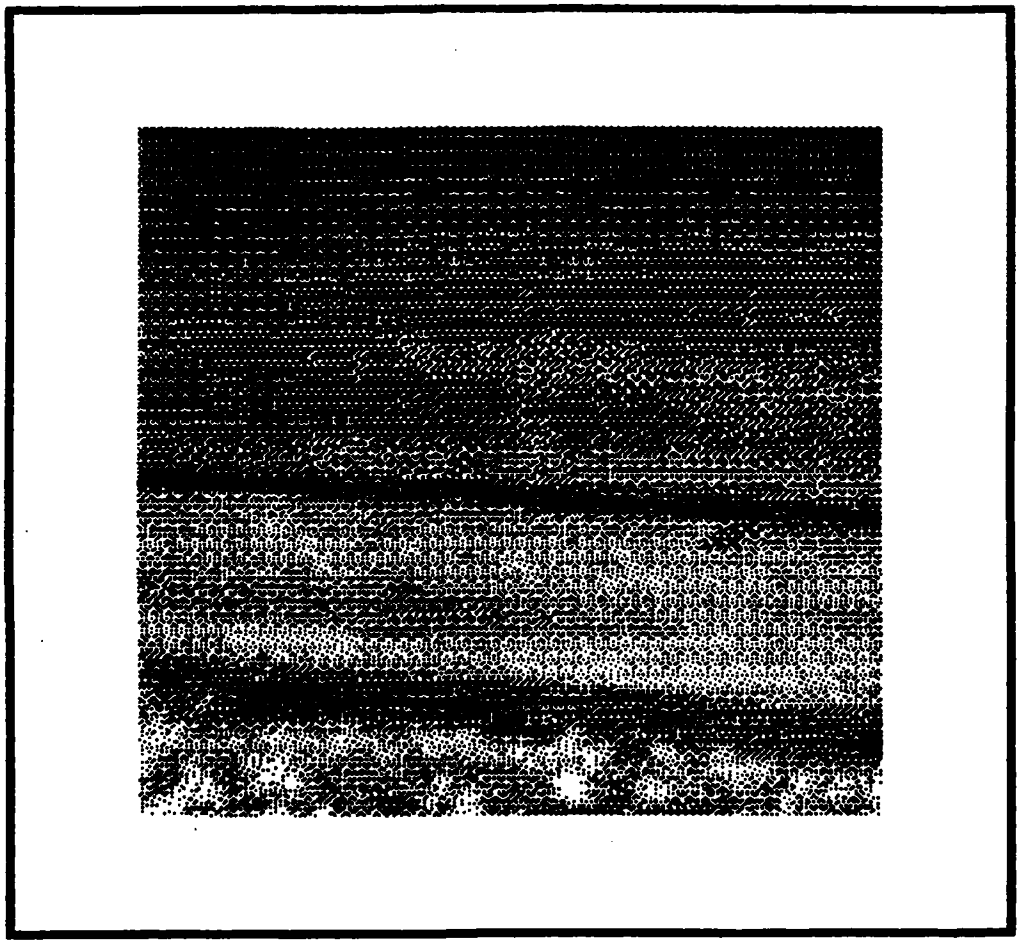
LOCALLY THRESHOLDED IMAG7.IR (7 X 7 NEIGHBORHOOD)
AFTER CONNECTIVITY TEST



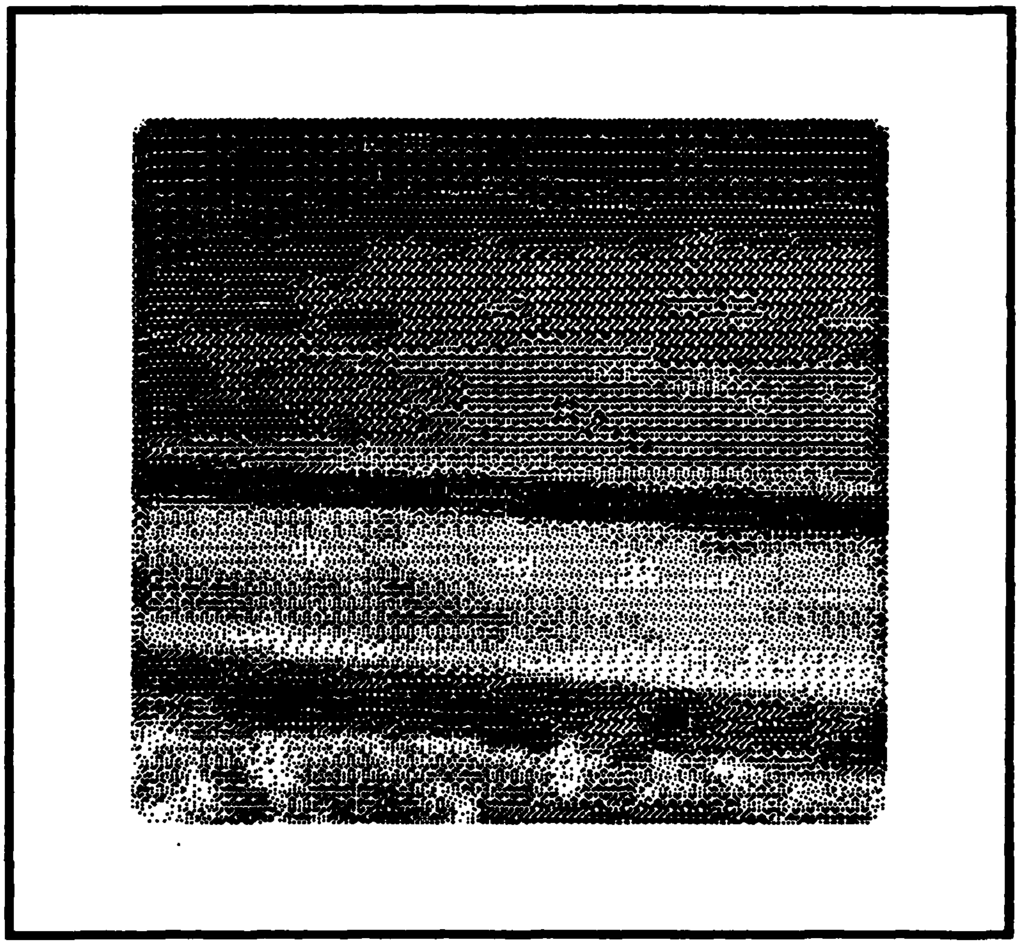
LOCALLY THRESHOLDED IMA07.IR (USING 7 X 7 NEIGHBORHOOD WITHOUT EDGING)



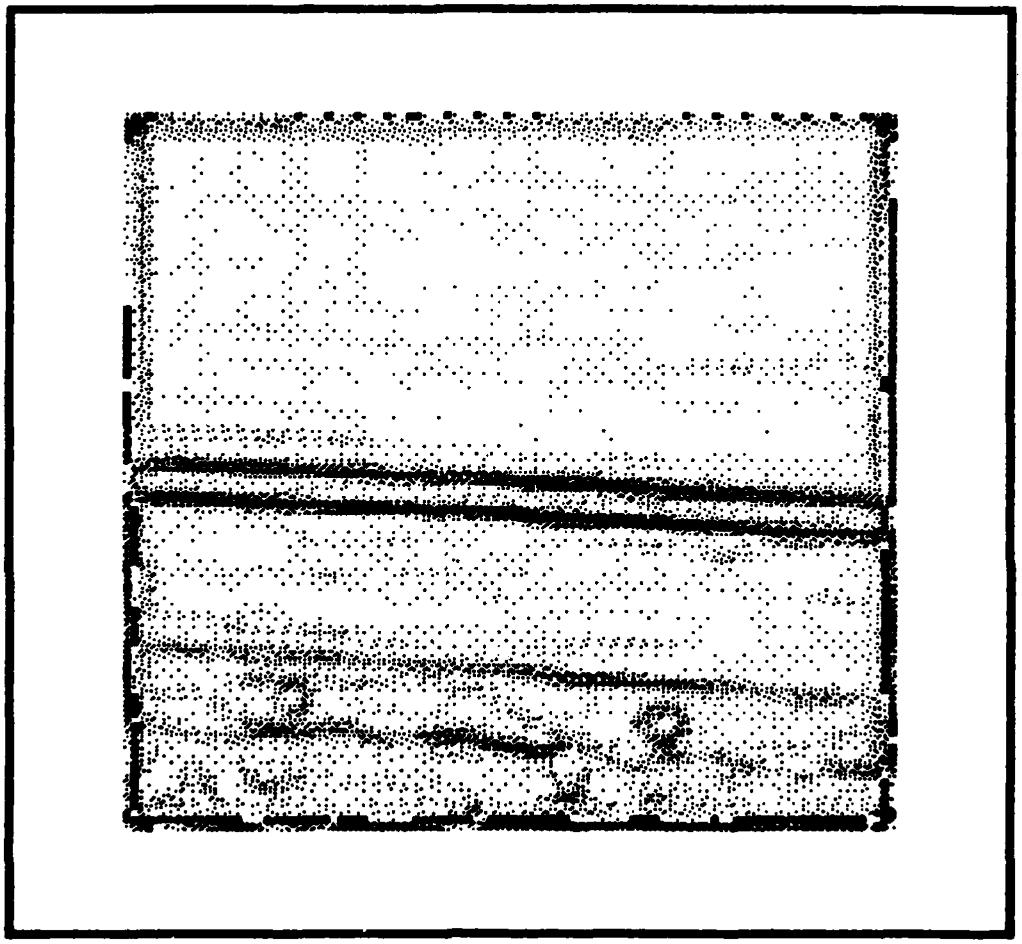
LOCALLY THRESHOLDED IMAG7.IR (USING 7 X 7 NEIGHBORHOOD WITHOUT EDGING) AFTER CONNECTIVITY TEST



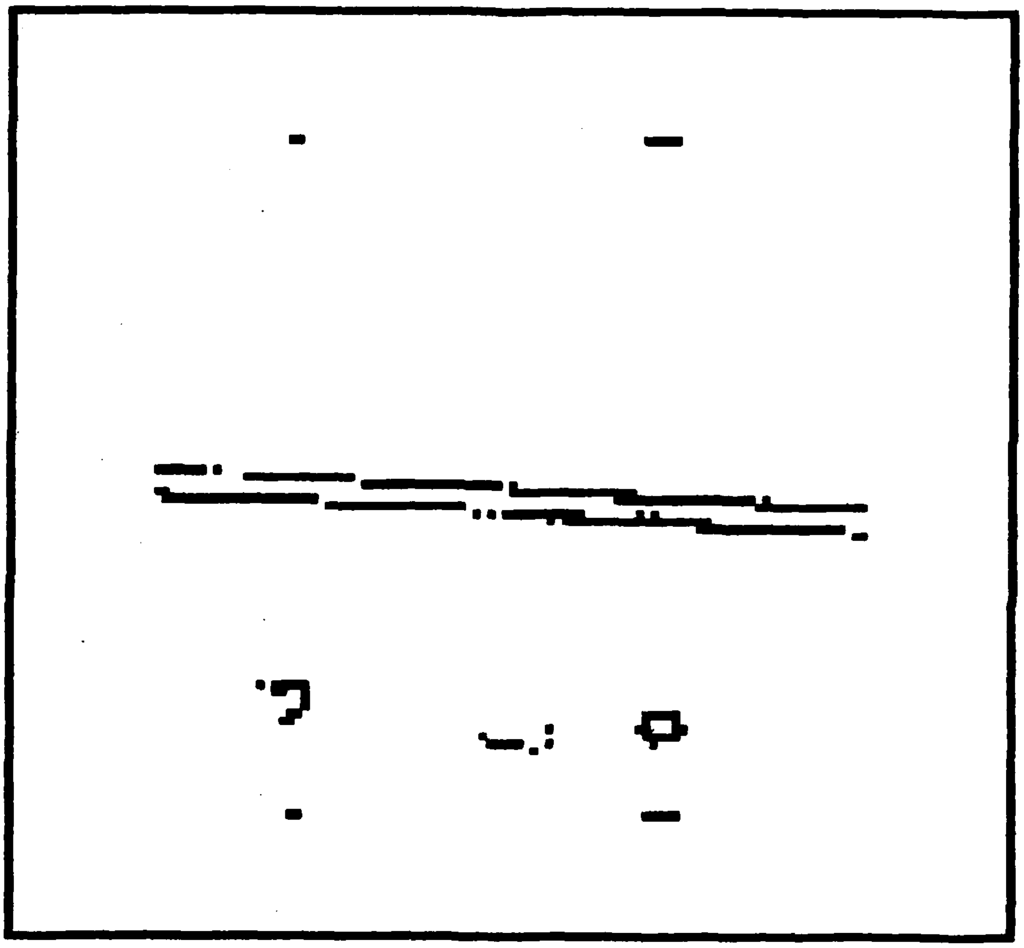
INFRARED IMAGE #8 (IMAG8.IR)



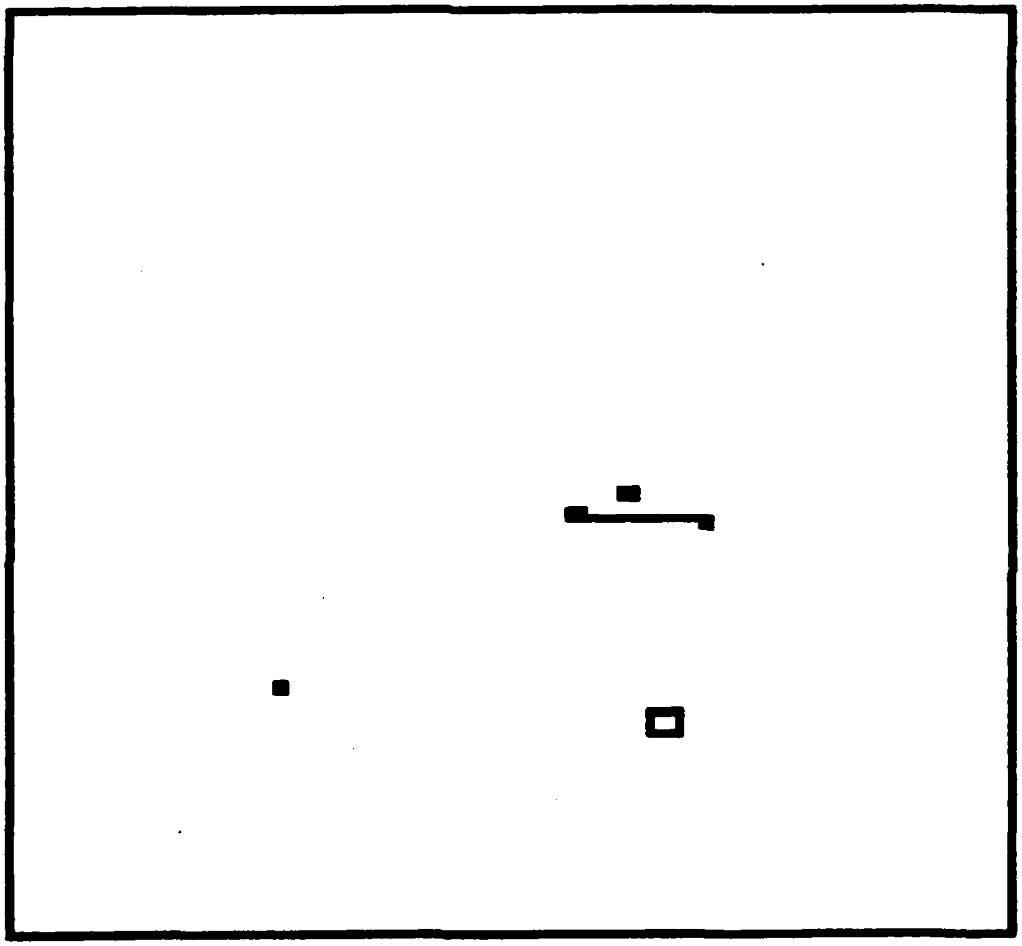
ENHANCED IMAGE OF IMAGB.IR



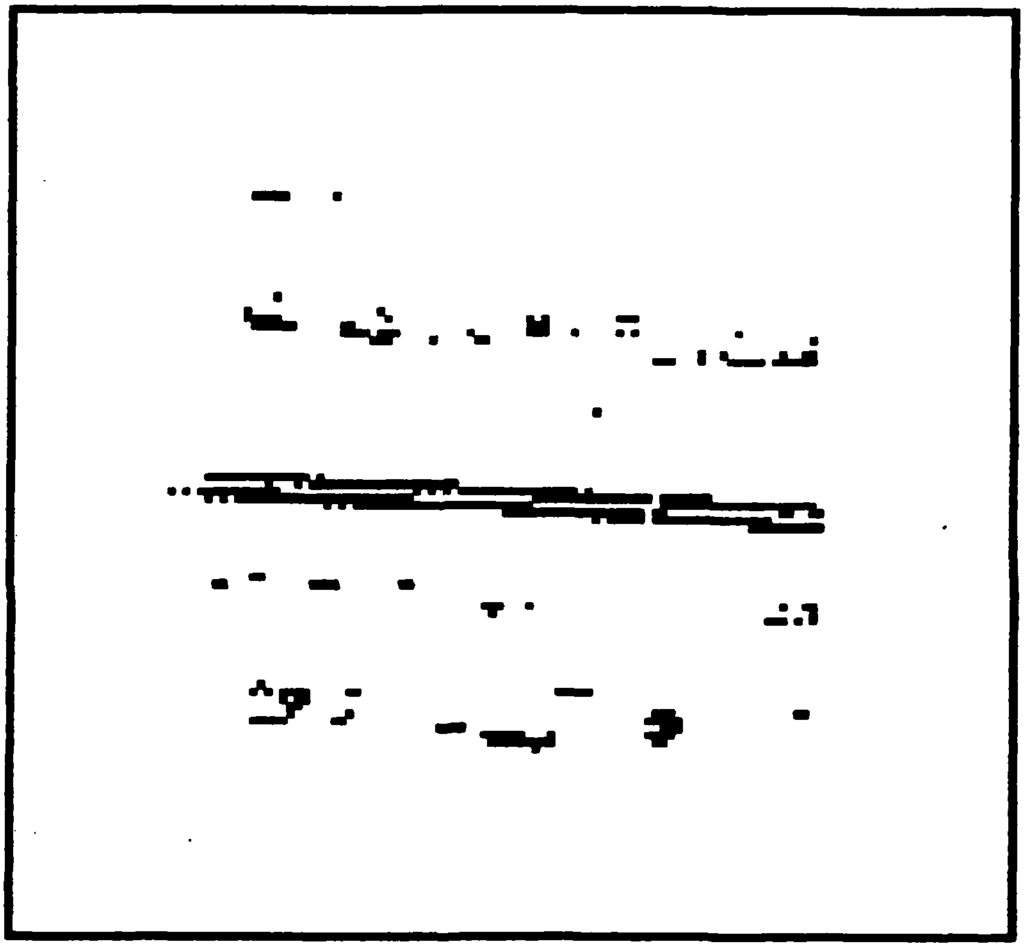
EDGED IMAGE OF IMAGB. IR



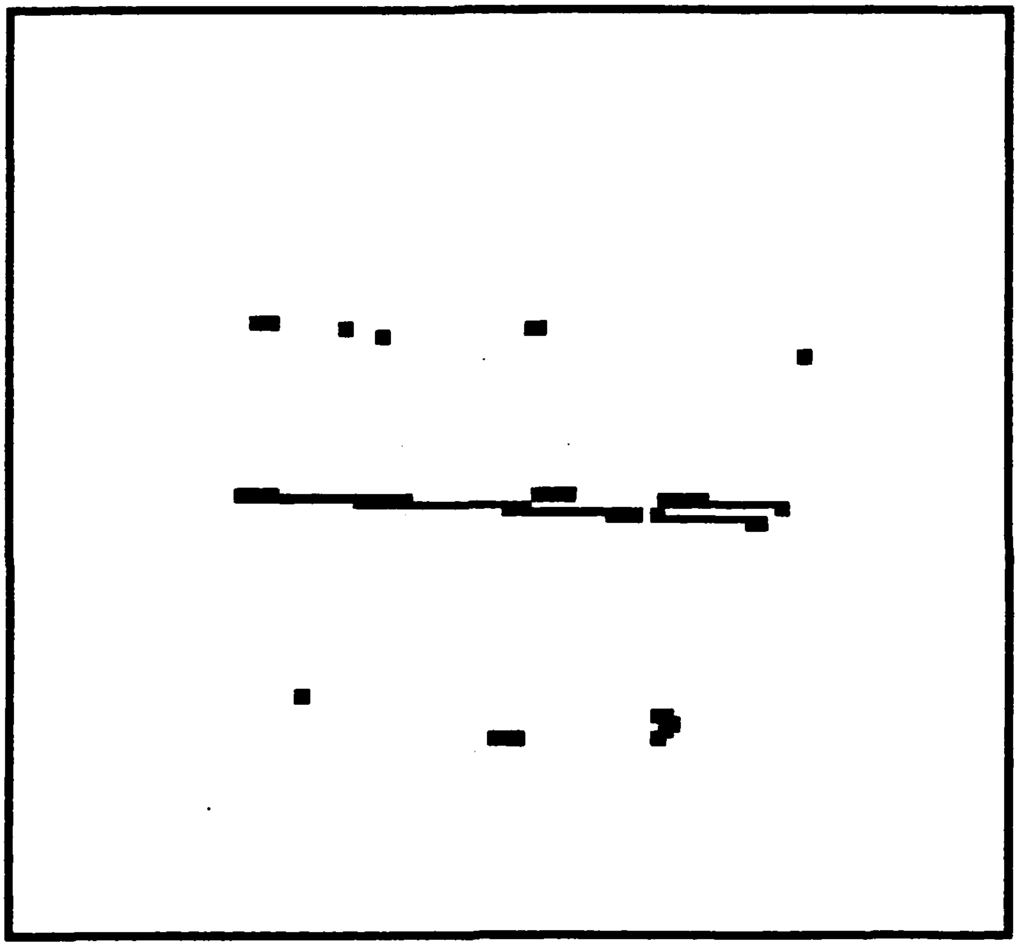
GLOBALLY THRESHOLDED IMAGE OF IMAG8.IR



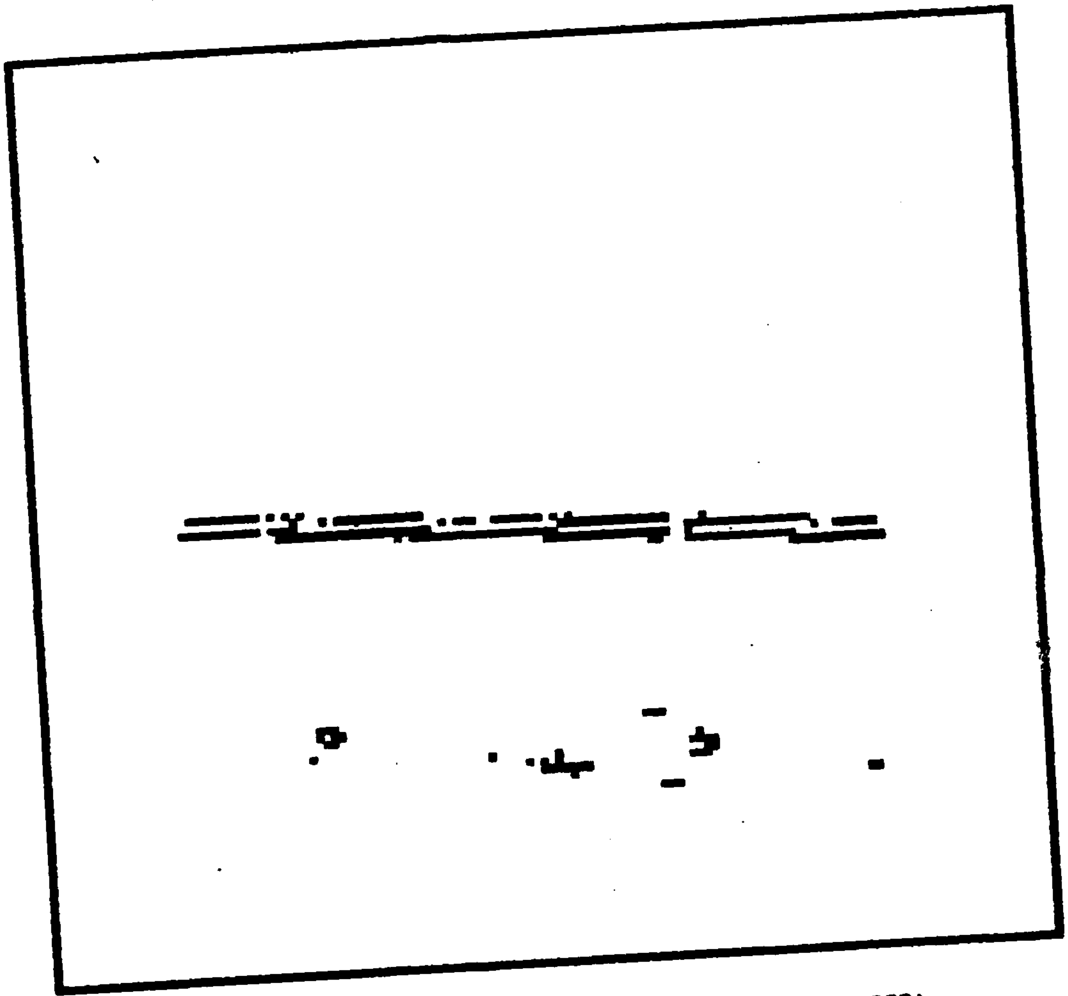
GLOBALLY THRESHOLDED IMAGE. IR AFTER CONNECTIVITY TEST



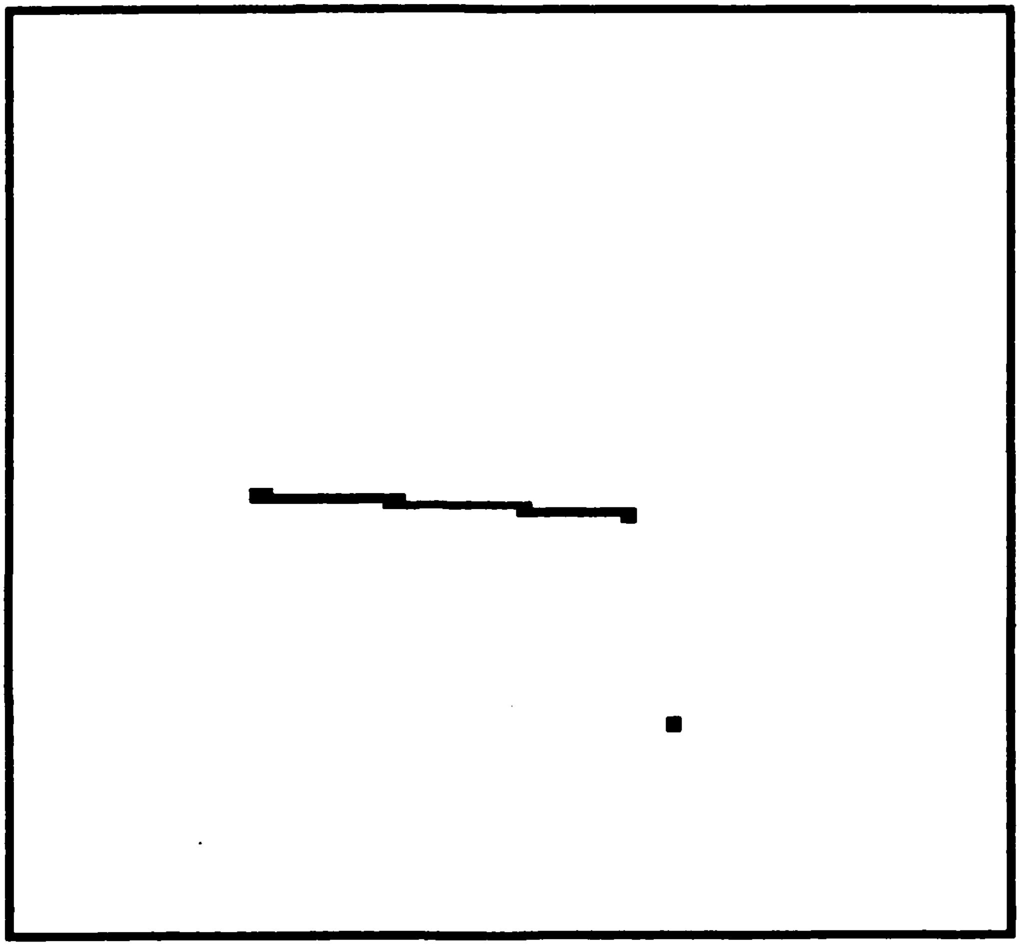
LOCALLY THRESHOLDED IMAGE. IR (17 X 17 NEIGHBORHOOD)



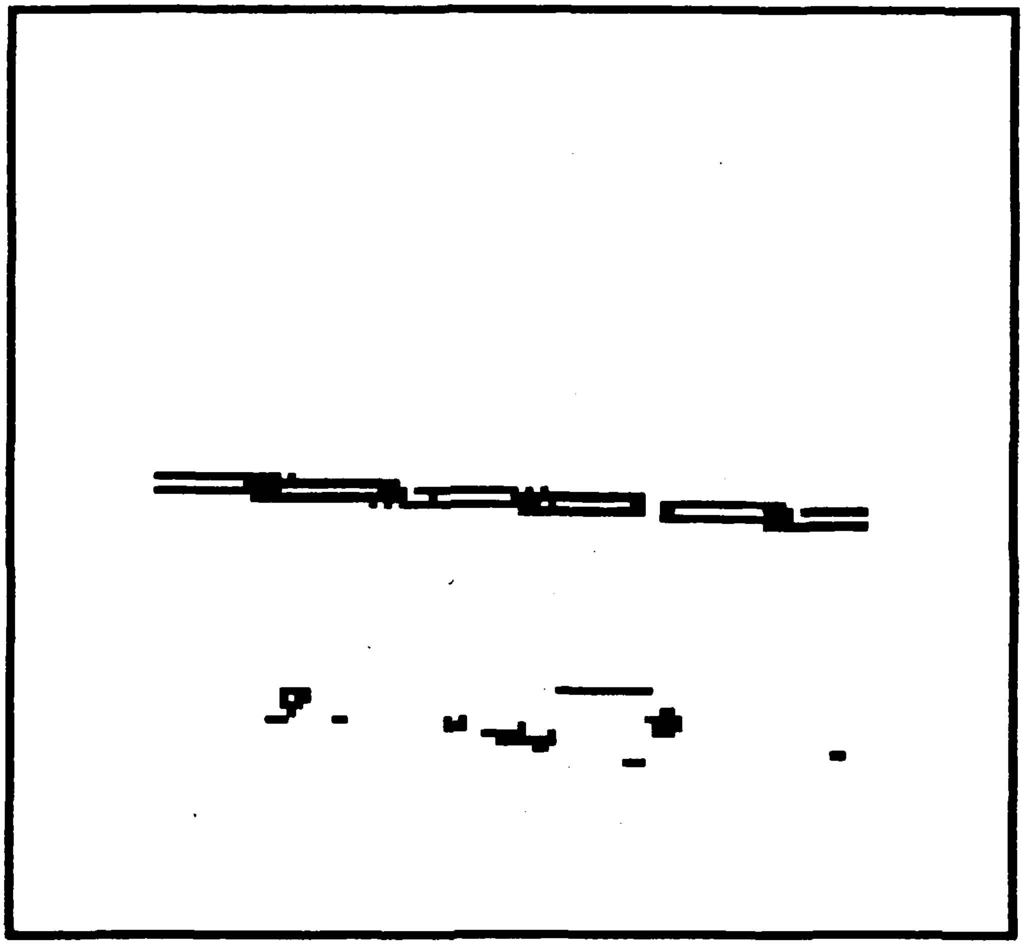
LOCALLY THRESHOLDED IMAGE.IR (17 X 17 NEIGHBORHOOD)
AFTER CONNECTIVITY TEST



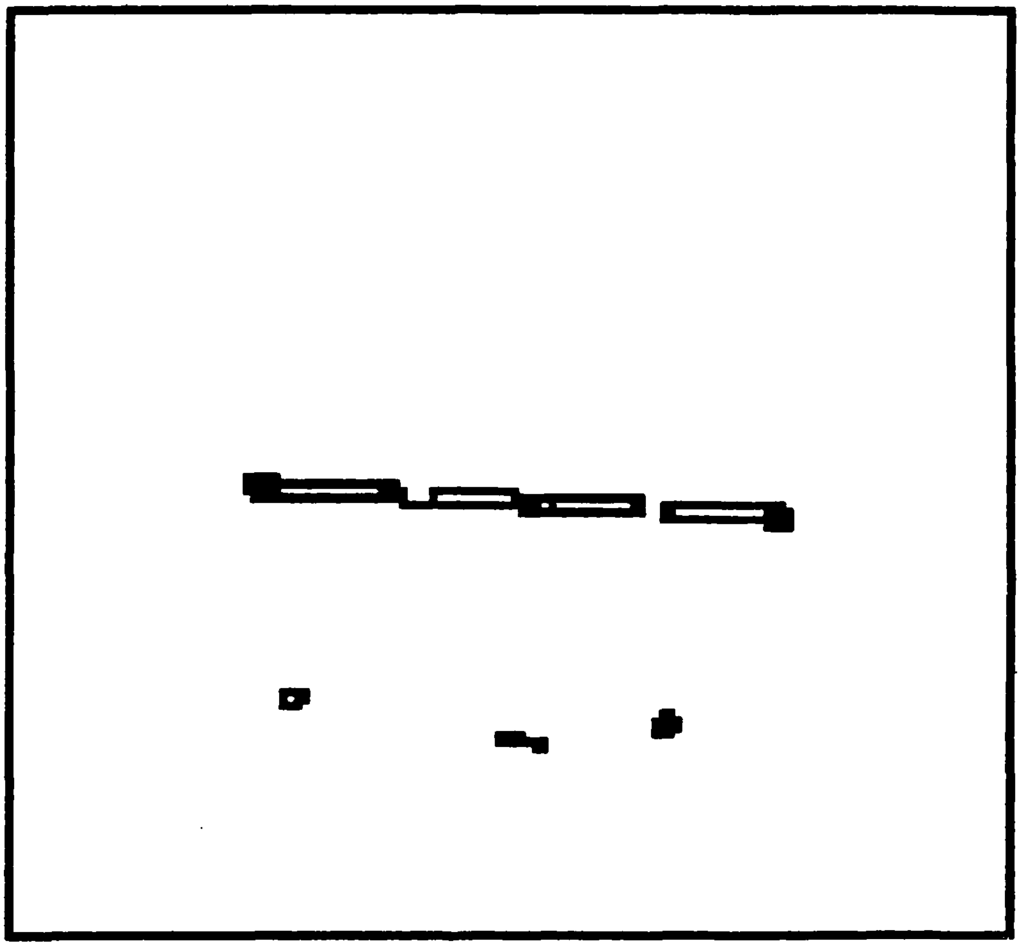
LOCALLY THRESHOLDED IMAGB.IR (7 X 7 NEIGHBORHOOD)



LOCALLY THRESHOLDED IMAGB. IR (7 X 7 NEIGHBORHOOD)
AFTER CONNECTIVITY TEST



LOCALLY THRESHOLDED IMAGE.IR (USING 7 X 7 NEIGHBORHOOD WITHOUT EDGING)



LOCALLY THRESHOLDED IMAGB. IR (USING 7 X 7 NEIGHBORHOOD
WITHOUT EDGING) AFTER CONNECTIVITY TEST

VITA

Robert D. Wells, born October 1, 1957, attended Auburn University in Auburn, Alabama for four years to be commissioned in the ROTC program with a B.S.E.E. degree in 1979. Then he served three years as a Digital Weapons Engineer for the Air Force Armament Laboratory at Eglin AFB, Florida, before entering the School of Engineering, Air Force Institute of Technology, in June, 1982.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; Distribution Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/EE/83D-72	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering; 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Techology Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) See block # 19	

12. PERSONAL AUTHOR(S) Robert D. Wells, B.S.E.E., Capt USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
MS Thesis	FROM ______ TO ______	1983 December	224

16. SUPPLEMENTARY NOTATION

Approved for public release: IAW AFR 190-17.

Lynn E. Wolaver 7 Feb 84

LYNN E. WOLAVER

Dean for Research and Professional Development

Air Force Institute of Technology (AFC)

Wright-Patterson AFB OH 45433

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	Pattern Recognition; Target Detection; Spatial Filtering
06	04		
17			

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

IMAGE FILTERING WITH BOOLEAN AND STATISTICAL OPERATORS

Thesis Advisor: Dr M. Kabrisky

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
Dr M. Kabrisky	513-255-5276	AFIT/ENG

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE

AFIT/GE/EE/83D-72

ABSTRACT

Edge extraction is an image processing technique for defining the edge information in an image. This effort researches different edging processes as applied to preprocessing for two pattern recognition processes. The first one is a cross-correlation method to find a target given that the target has a known size, orientation, and aspect. Correlation is performed in the spatial frequency domain with two-dimensional fast Fourier transforms of the searched edge image and a hand drawn edge template to correct for translation only.

The second pattern recognition process researched also uses edging as one step of a purely spatial domain algorithm. The approach locates targets in infrared images that can be described a "hot" clusters. A cluster recognition algorithm by Hamadani is implemented and altered for testing of local thresholding and thresholding rules. The algorithm is shown to be effective on real infrared images provided by the thesis sponsor, the Air Force Armament Laboratory at Eglin AFB, Florida.

END

FILMED

3-84

DTIC